740 Family

740 Family Sample Programs Collection

Contents

1. Distinctive Features of 740 Family Instruction Set

The 740 Family instruction set offers the following distinctive features.

(1) An efficient instruction set with many addressing modes allowing the effective use of user program area.
(2) The same bit set/clear, bit test, and branch instructions can be applied for Accumulator, Memory and I/O area.
(3) Multiple interrupts allow a variety of servicing of periodic or non-periodic events.
(4) Powerful indexed addressing performs various byte and table-reference operations.
(5) Decimal mode does not require any software correction for decimal operations.
(6) Accumulator does not need to be used in operations using memories and/or I/Os.

2. Effective use of 740 Family MCU Distinctive Instructions

2.1 Memory-to-Memory Operations

(1) Addition (mnemonic ADC)

When X Modified Operation Mode Flag T is "1", the ADC instruction adds the contents of M (X), Memory M and Carry Flag C and stores the results in M (X) and Carry Flag C. At this point, the contents of Accumulator A remain the same, but the status flags are changed. In this case, M (X) represents the contents of the memory at the address indicated by Index Register X.

Example:

```
CLC                 (a)
SET                 (b)
LDX     #ADDATA     (c)
ADC     $10, X      (d)
CLT                 (e)
```

Explanation:

(a) Set Carry Flag C to "0".

(b) Set X Modified Operation Mode Flag T to "1".

(c) Load #ADDATA (example: 70H) to Index Register X.

(d) Add the contents of the memory at address 70H, contents of address 80H (70H + 10H) and contents of Carry Flag C; store the results in address 70H and Carry Flag C.

(e) Set X Modified Operation Mode Flag T to "0".

For example, if the contents of address 70H are 53H and the contents of address 80H are 21H, the contents of address 70H will be 74H and Carry Flag C will be "0" after execution of the instructions in steps (a) to (e).

(2) Subtraction (mnemonic SBC)

When X Modified Operation Mode Flag T is "1", the SBC instruction subtracts the value of Memory M and the complement of Carry Flag C from the contents of M (X), and stores the results in M (X) and Carry Flag C. At this point, the contents of Accumulator A remain the same, but the status flags are changed. In this case, M (X) represents the contents of the memory at the address indicated by Index Register X.

Example:

```
SEC                 (a)
SET                 (b)
LDX     #SBDATA     (c)
SBC     $10, X      (d)
CLT                 (e)
```

Explanation:

(a) Set Carry Flag C to "1".

(b) Set X Modified Operation Mode Flag T to "1".

(c) Load #SBDATA (example: $80) to Index Register X.

(d) Subtract the contents of address 90H (80H + 10H) and the complement of Carry Flag C from the contents of the memory at address 80H; store the results in address 80H and Carry Flag C.

(e) Set X Modified Operation Mode Flag T to "0".

For example, if the contents of address 80H are 53H and the contents of address 90H are 21H, the contents of address 80H will be 32H and Carry Flag C will be "1" after execution of the instructions in steps (a) to (e).

2.2 Bit Branch Instructions

(1) Branch On Bit Set (mnemonic BBS)

The BBS instruction tests the designated bit i of Accumulator A or Memory M. If the bit is “1”, the program branches to the specified address. The branch address is specified by a relative address.

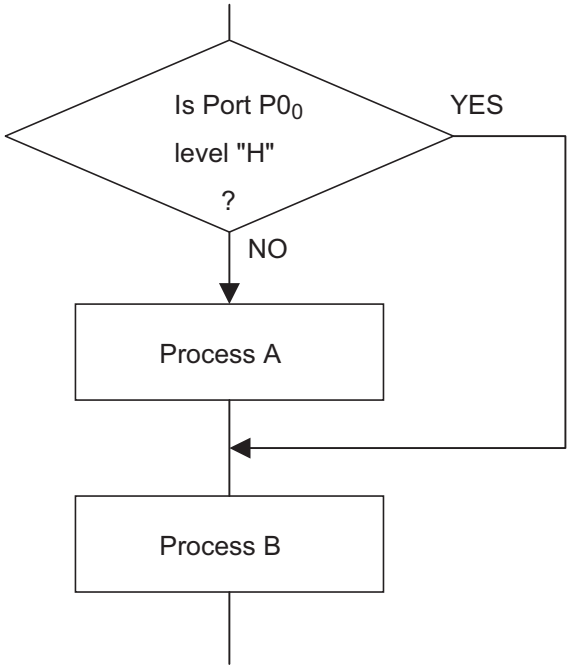
If the bit is “0”, the next instruction is executed.

Example:

```
        BBS     0 , P 0 , START
        Process A
START:
        Process B
```

Explanation:

If bit 0 of Port P0 is “1”, the program branches to START. If bit 0 of Port P0 is “0”, the program continues on the next instruction.



(2) Branch On Bit Clear (mnemonic BBC)

The BBC instruction tests the designated bit i of Accumulator A or Memory M. If the bit is “0”, the program branches to the specified address. The branch address is specified by a relative address.

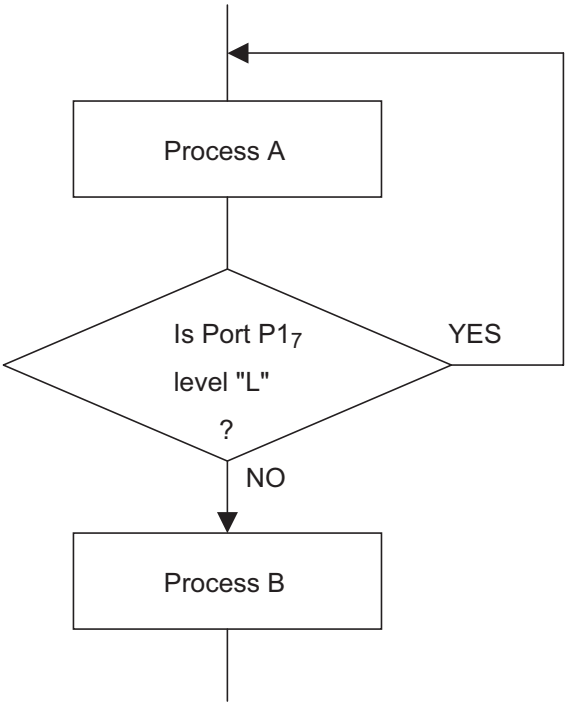
If the bit is “1”, the next instruction is executed.

Example:

```
        RESET:
                Process A
                BBC     7, P1, RESET
                Process B
```

Explanation:

If bit 7 of Port P1 is “0”, the program branches to RESET. If bit 7 of Port P1 is “1”, the program continues on the next instruction.



2.3 Bit Managing (Set/Reset)

(1) Set Bit (mnemonic SEB)

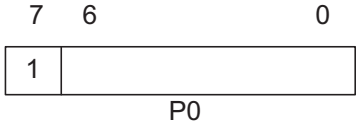
The SEB instruction sets the contents of the designated bit i of Accumulator A or Memory M to “1”.

Example:

```
SEB     7, P0
```

Explanation:

The contents of bit 7 of Port P0 are set to “1”.



For example, if the contents of Port P0 are 53H, the contents will be D3H after the instruction is executed.

(2) Clear Bit (mnemonic CLB)

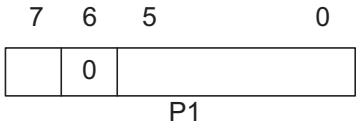
The CLB instruction sets the contents of the designated bit i of Accumulator A or Memory M to “0”.

Example:

```
CLB     6, P1
```

Explanation:

The contents of bit 6 of Port P1 are set to “0”.



For example, if the contents of Port P1 are 53H, the contents will be 13H after the instruction is executed.

2.4 Rotate Shift

(1) Rotate One Bit Left (mnemonic ROL)

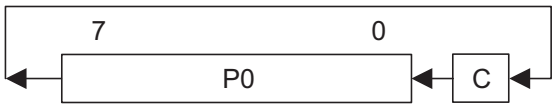
The ROL instruction puts the contents of Accumulator A or Memory M and Carry Flag C together as a 9-bit row and rotates the contents one bit to the left. Then, the contents of Carry Flag C are stored in bit 0 of Accumulator A or Memory M, and the contents of bit 7 of Accumulator A or Memory M are stored in Carry Flag C.

Example:

```
ROL     P0
```

Explanation:

Port P0 is connected to Carry Flag C and their contents are rotated one bit to the left.



For example, if the contents of Port P0 are 53H and Carry Flag C is “1”, the contents of Port P0 will be A7H and Carry Flag C will be “0” after the instruction is executed.

(2) Arithmetic Shift Left (mnemonic ASL)

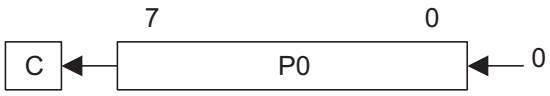
The ASL instruction shifts all bits of Accumulator A or Memory M one bit to the left. In this case, bit 0 of Accumulator A or Memory M will be “0”, and the contents of bit 7 of Accumulator A or Memory M are stored in Carry Flag C.

Example:

```
ASL     P0
```

Explanation:

All Port P0 bits are shifted one bit to the left.



For example, if the contents of Port P0 are 53H and Carry Flag C is “1”, the contents of Port P0 will be A6H and Carry Flag C will be “0” after the instruction is executed.

(3) Rotate One Bit Right (mnemonic ROR)

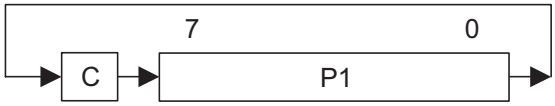
The ROR instruction puts the contents of Accumulator A or Memory M and Carry Flag C together as a 9-bit row and rotates the contents one bit to the right. Then, the contents of Carry Flag C are stored in bit 7 of Accumulator A or Memory M, and the contents of bit 0 of Accumulator A or Memory M are stored in Carry Flag C.

Example:

```
ROR     P1
```

Explanation:

Port P1 and Carry Flag C are connected and their contents are rotated one bit to the right.



For example, if the contents of Port P1 are 53H and Carry Flag C is “1”, the contents of Port P1 will be 29H and Carry Flag C will remain as “1” after the instruction is executed.

(4) Logical Shift Right (mnemonic LSR)

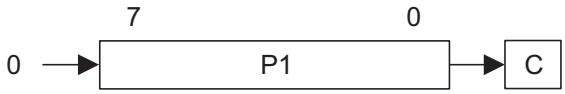
The LSR instruction shifts all bits of Accumulator A or Memory M one bit to the right. In this case, bit 7 of Accumulator A or Memory M will be “0”, and the contents of bit 0 of Accumulator A or Memory M are stored in Carry Flag C.

Example:

```
LSR     P1
```

Explanation:

All bits of Port P1 are shifted one bit to the right.



For example, if the contents of Port P1 are 53H and Carry Flag C is “1”, the contents of Port P1 will be 29H and Carry Flag C will remain as “1” after the instruction is executed.

(5) Rotate Right of Four Bits (mnemonic RRF)

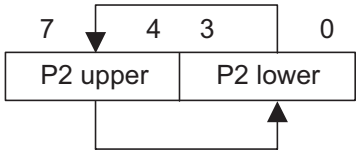
The RRF instruction rotates the contents of Memory M four bits to the right. As a result, the contents of the upper-order and lower-order four bits of Memory M are reversed, but the order of the respective four bits does not change.

Example:

```
RRF     P2
```

Explanation:

The contents of the upper-order and lower-order four bits of Port P2 are reversed.



For example, if the contents of Port P2 are 53H, the contents will be 35H after the instruction is executed.

3. Basic Processing Program Example

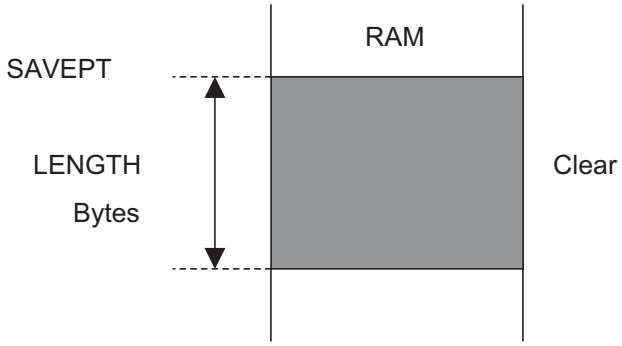
3.1 RAM Clear

(1) Description

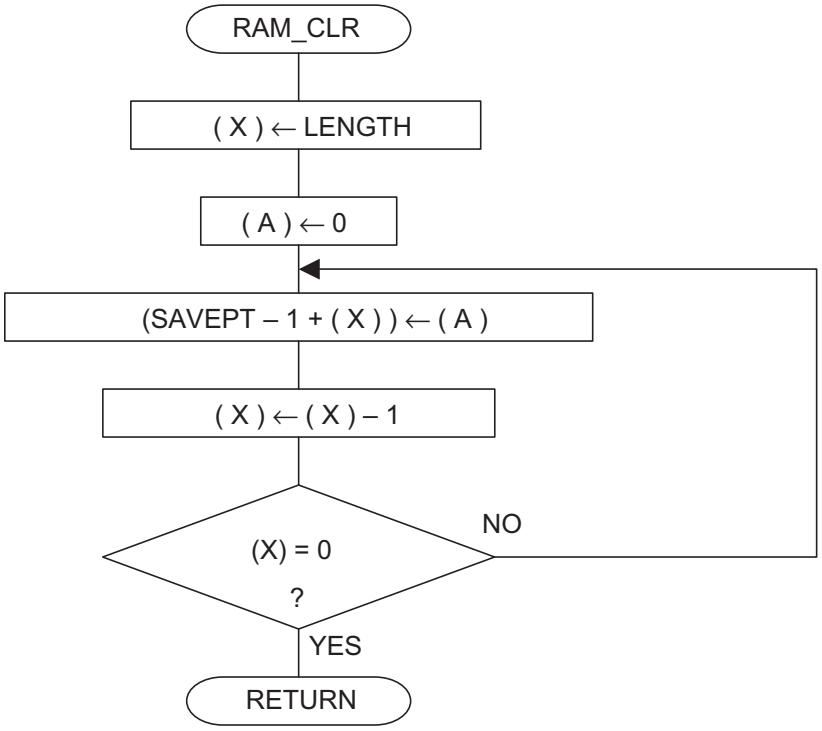
This program sets the area and clears the RAM.

(2) Explanation

The program clears LENGTH Bytes of RAM from RAM address SAVEPT.



(3) Flowchart



(4) Program List

```
;*******************************************************
;
;            RAM clear routine
;
;*******************************************************
;
RAM_CLR:
            LDX                 #LENGTH  ;RAM length
            LDA    #$00
RAMCL1:     STA    SAVEPT-1,X   ;Clear from SAVEPT
            DEX                 ; -to SAVEPT+LENGTH-1
            BNE                 RAMCL1   ;Clear end ?
            RTS                 ;Yes
```

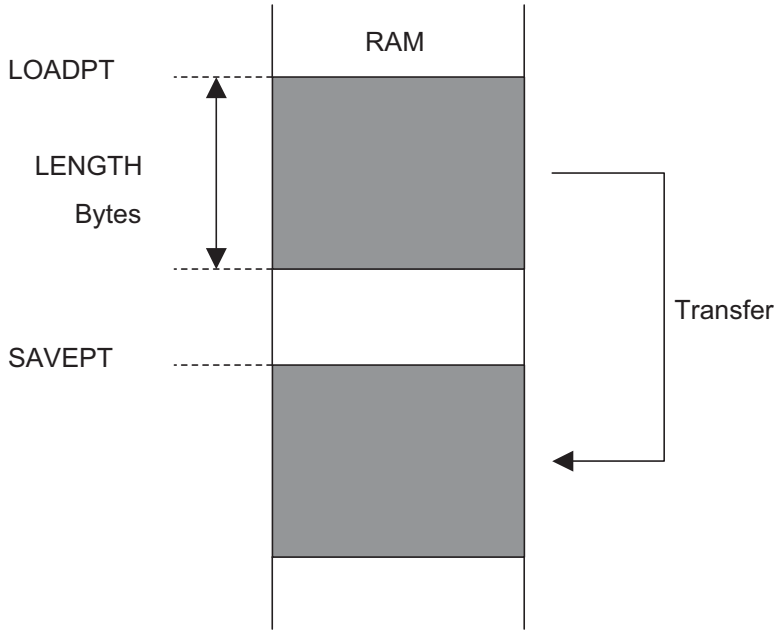
3.2 Data Transfer (RAM)

(1) Description

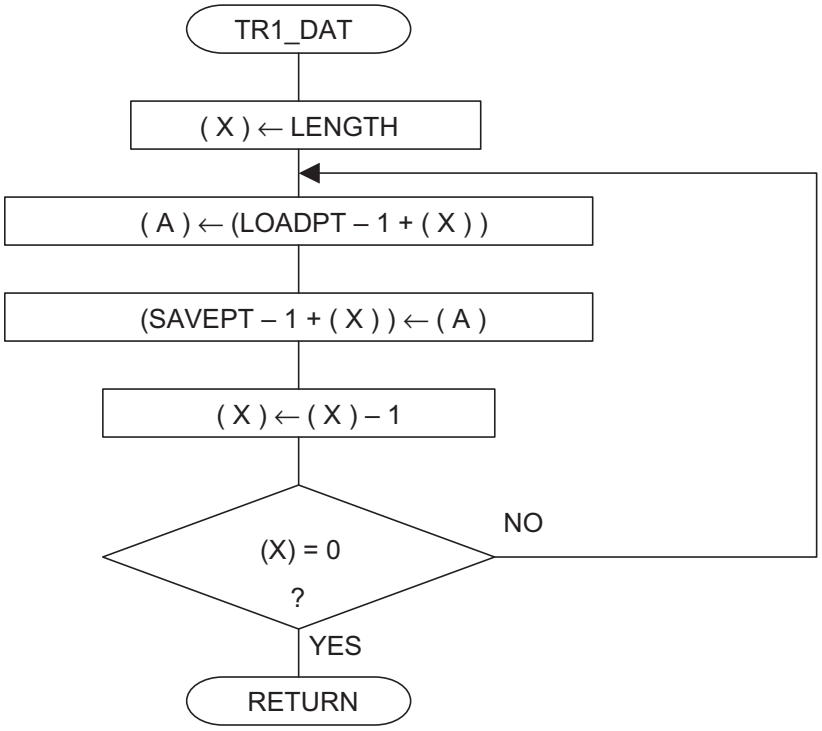
Data is transferred within the RAM area.

(2) Explanation

LENGTH Bytes of data are transferred from RAM address LOADPT to addresses starting at SAVEPT.



(3) Flowchart



(4) Program List

```
;******************************************************
;
;            RAM data transfer routine
;
;******************************************************
;
TR1_DAT:
          LDX       #LENGTH         ;RAM length
TR1_01:   LDA       LOADPT-1,X      ;Transfer data from LOADPT
          STA       SAVEPT-1,X      ; -to SAVEPT
          DEX
          BNE       TR1_01          ;Transfer end ?
          RTS                       ;Yes
```

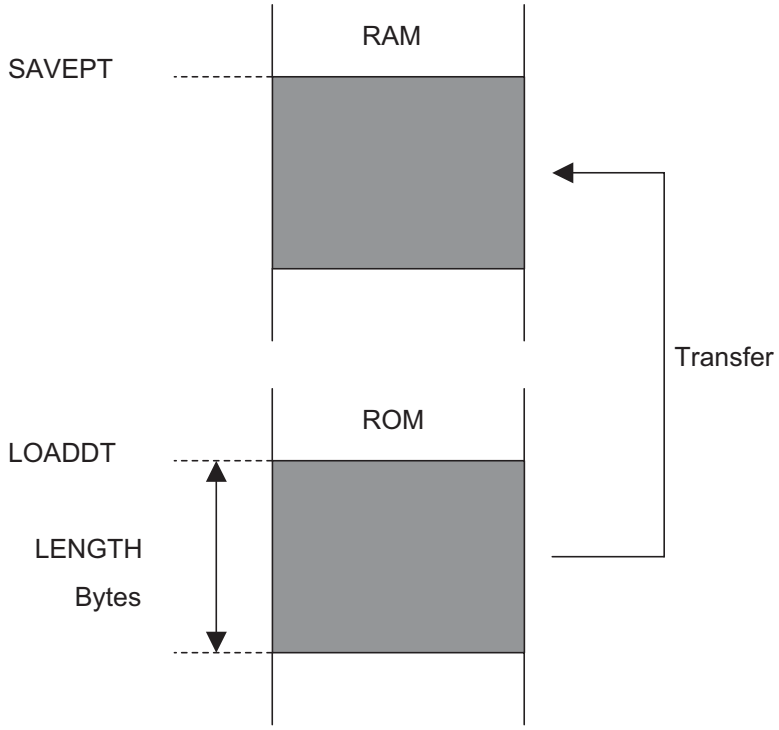
3.3 Data Transfer (ROM address fixed)

(1) Description

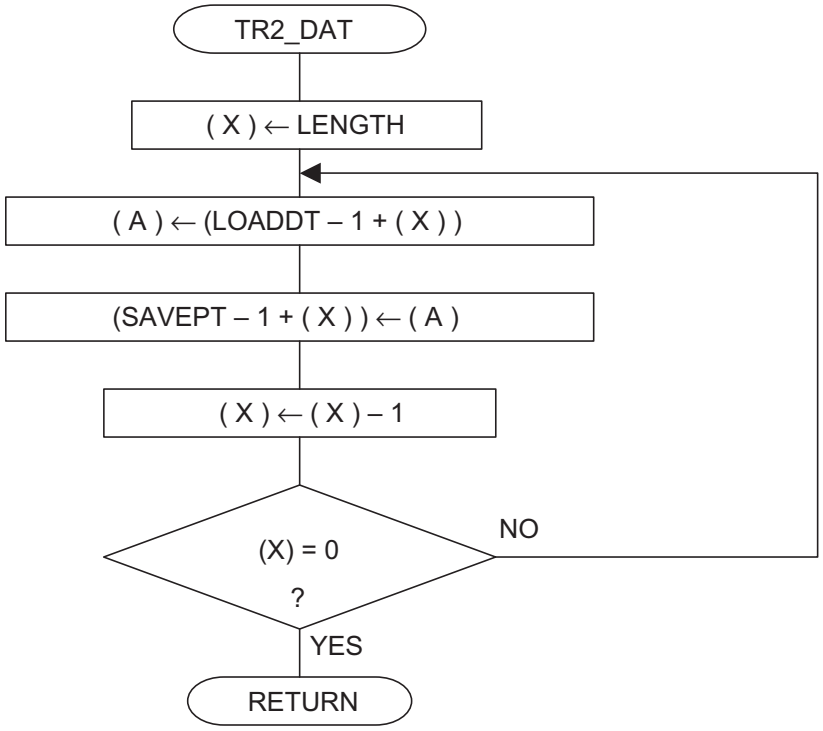
Data (address fixed) is transferred from the ROM area.

(2) Explanation

LENGTH Bytes of data are transferred from ROM address LOADDT to continuous RAM addresses, starting at SAVEPT.



(3) Flowchart



(4) Program List

```
;*******************************************************
;
;       ROM data transfer routine(address fixed)
;
;*******************************************************
;
TR2_DAT:
        LDX   #LENGTH         ;ROM length
TR2_01: LDA   LOADDT-1,X      ;Transfer data from LOADDT
        STA   SAVEPT-1,X      ; -to SAVEPT
        DEX
        BNE   TR2_01          ;Transfer end ?
        RTS                   ;Yes
```

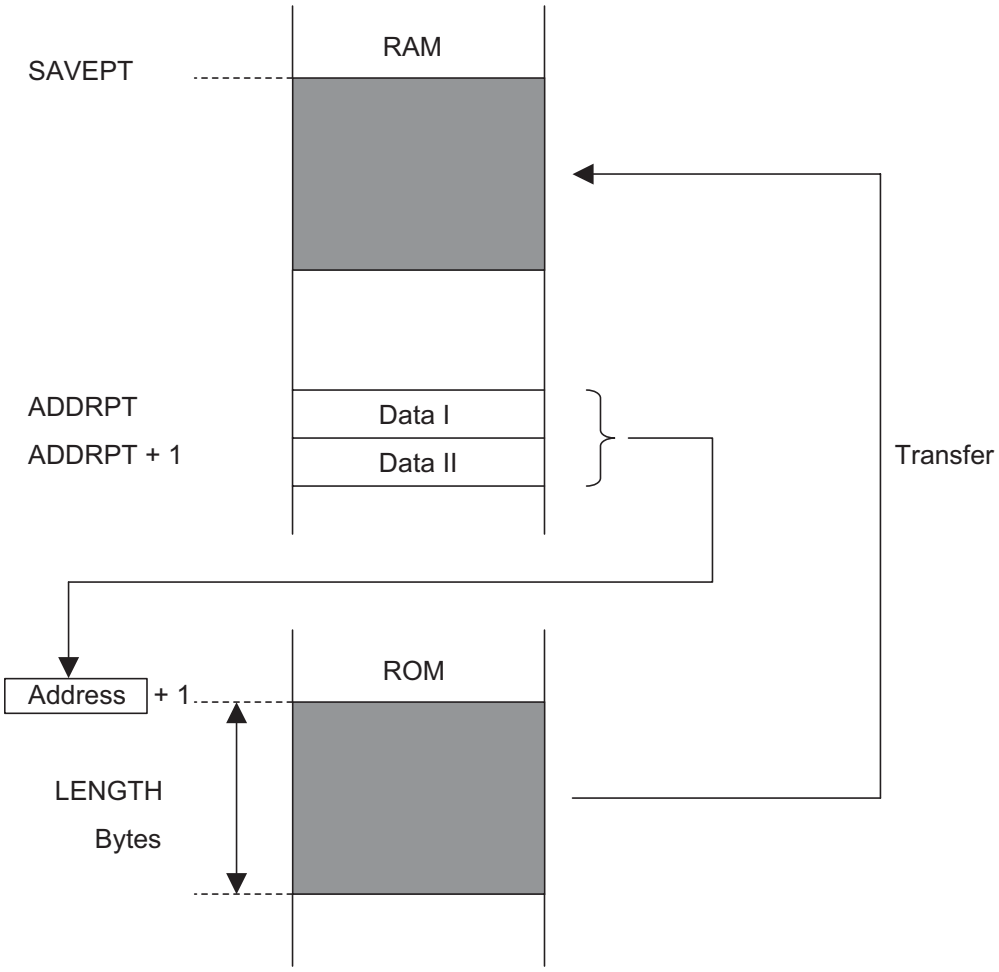
3.4 Data Transfer (ROM address variable)

(1) Description

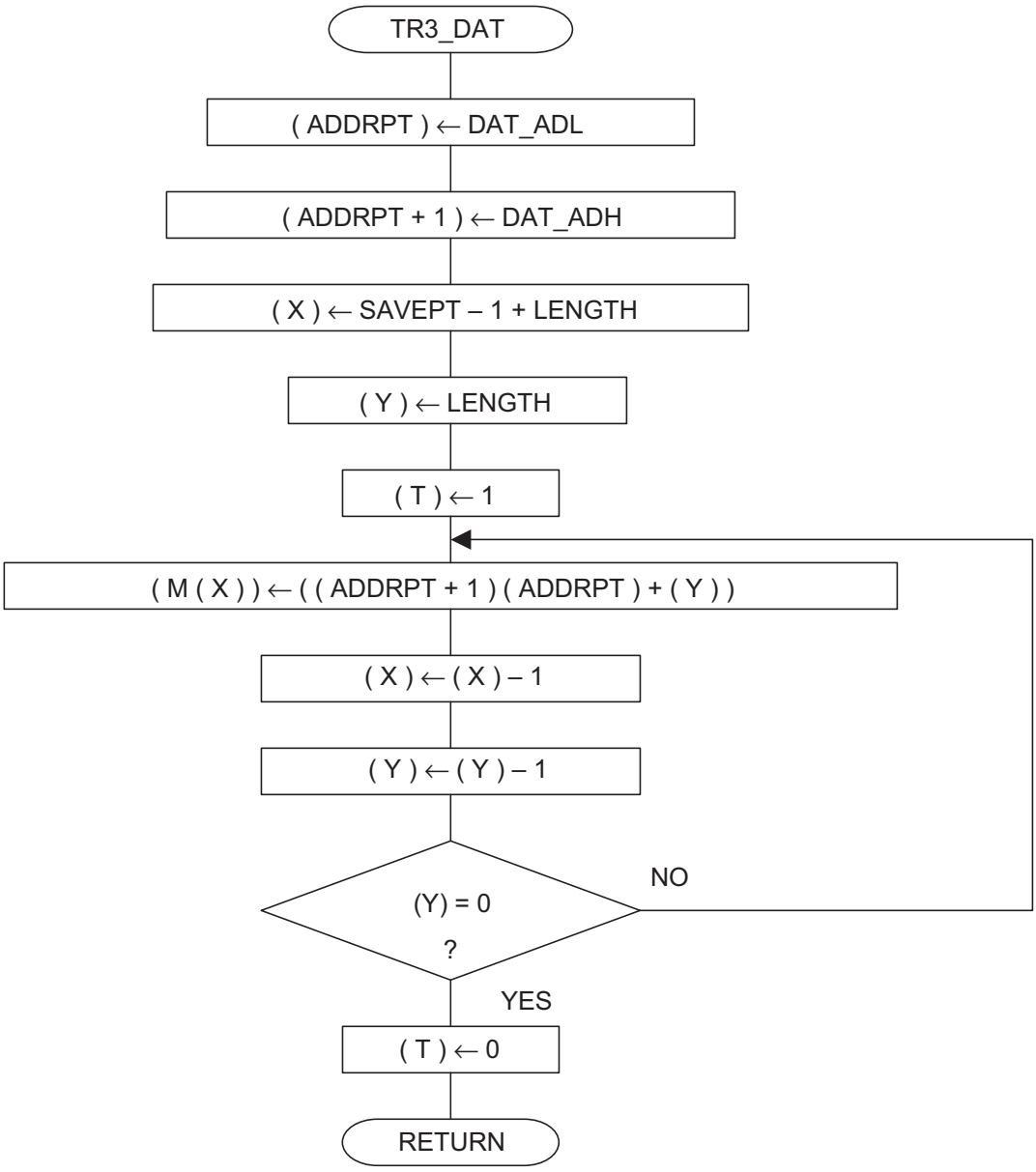
Data (address variable) is transferred from the ROM area.

(2) Explanation

LENGTH Bytes of data are transferred to continuous RAM addresses, starting at SAVEPT, from the ROM address +1 specified in the contents of RAM address ADDRPT +1 and ADDRPT.



(3) Flowchart



Note: M (X) represents the memory of the address indicated by Index Register X.

(4) Program List

```
;*******************************************************
;
;       ROM data transfer routine(address float)
;
;*******************************************************
;
TR3_DAT:
        LDM     #DAT_ADL,ADDRPT
        LDM     #DAT_ADH,ADDRPT+1
;------------------------------
        LDX     #SAVEPT-1+LENGTH
        LDY     #LENGTH         ;ROM length
        SET                     ;Transfer data from
TR3_01: LDA     (ADDRPT),Y      ; -(ADDRPT+1)(ADDRPT)+1
        DEX                     ; -to SAVEPT
        DEY                     ;
        BNE     TR3_01          ;Transfer end ?
        CLT                     ;Yes
        RTS
```

3.5 Data Sort

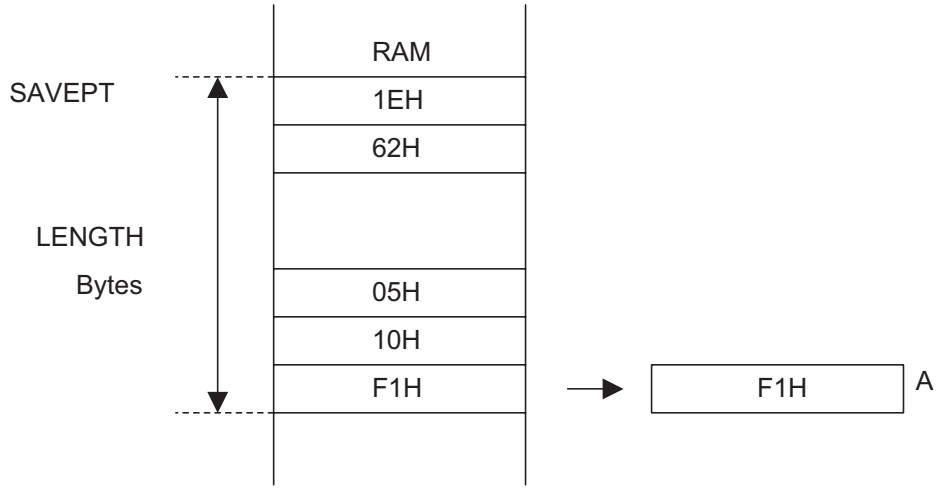
(1) Description

The data in the RAM area is sorted in descending order.

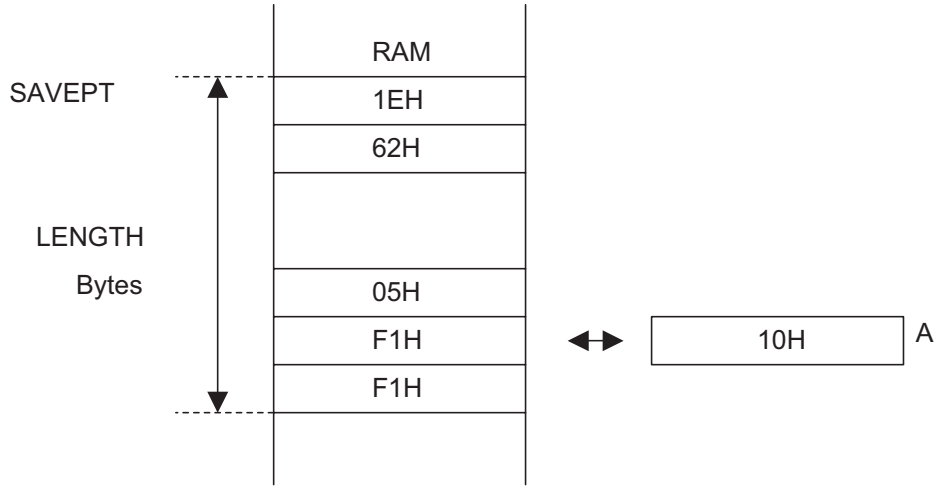
(2) Explanation

LENGTH bytes of data are sorted in descending order from RAM address SAVEPT.

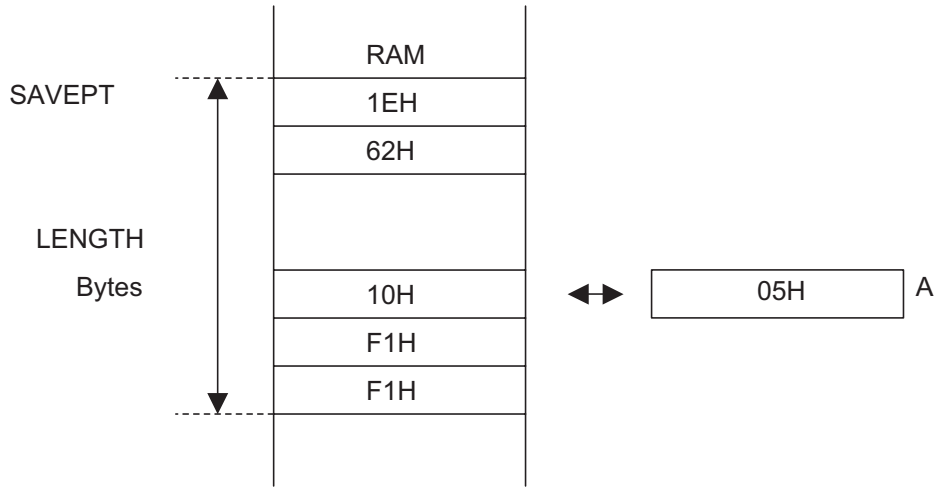
(a) First, the memory contents of the highest address to be sorted are stored in Accumulator A.



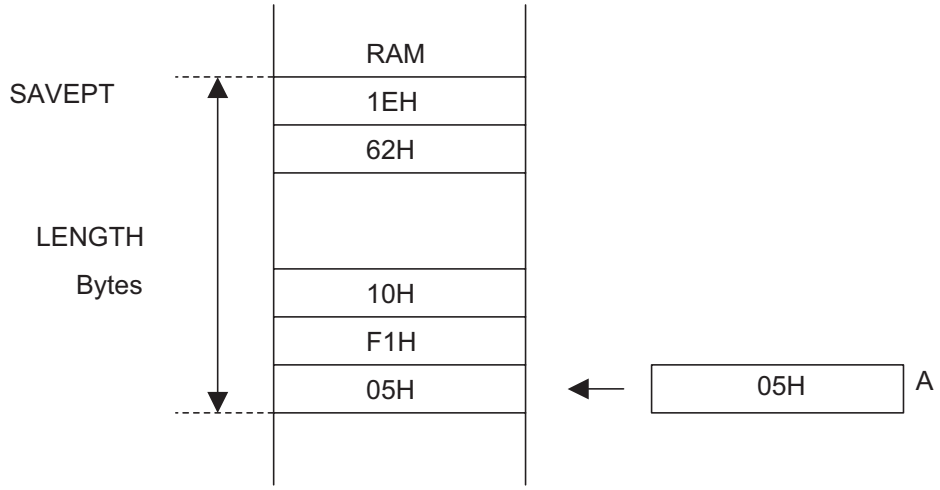
(b) Next, the memory contents of the next lower address are compared with the contents of Accumulator A. At this time, if the contents of Accumulator A are equal to or larger than the contents of the lower address, the contents of the memory and the Accumulator A are switched.



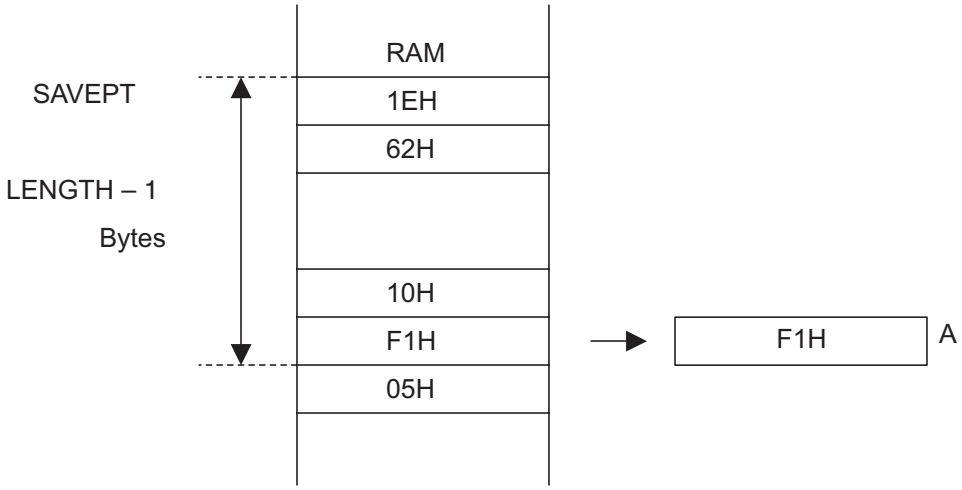
(c) The contents of the next lower address are compared with the contents of Accumulator A. At this time, if the contents of Accumulator A are equal to or larger than the contents of the memory, the contents of the memory and the Accumulator A are switched again.



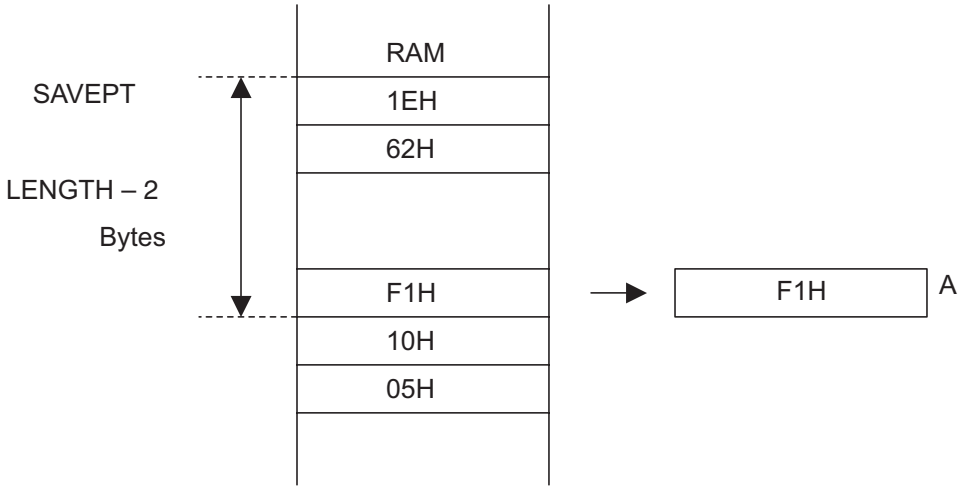
(d) This process is repeated until it reaches the lowest address and the contents of its memory are sorted, after which the lowest value will be stored in Accumulator A. The contents of this Accumulator A will be sent to the highest address in the memory.



(e) Steps (a) through (d) are executed on the sorted area of which the highest address has been decreased, (LENGTH bytes - 1 byte) from SAVEPT.



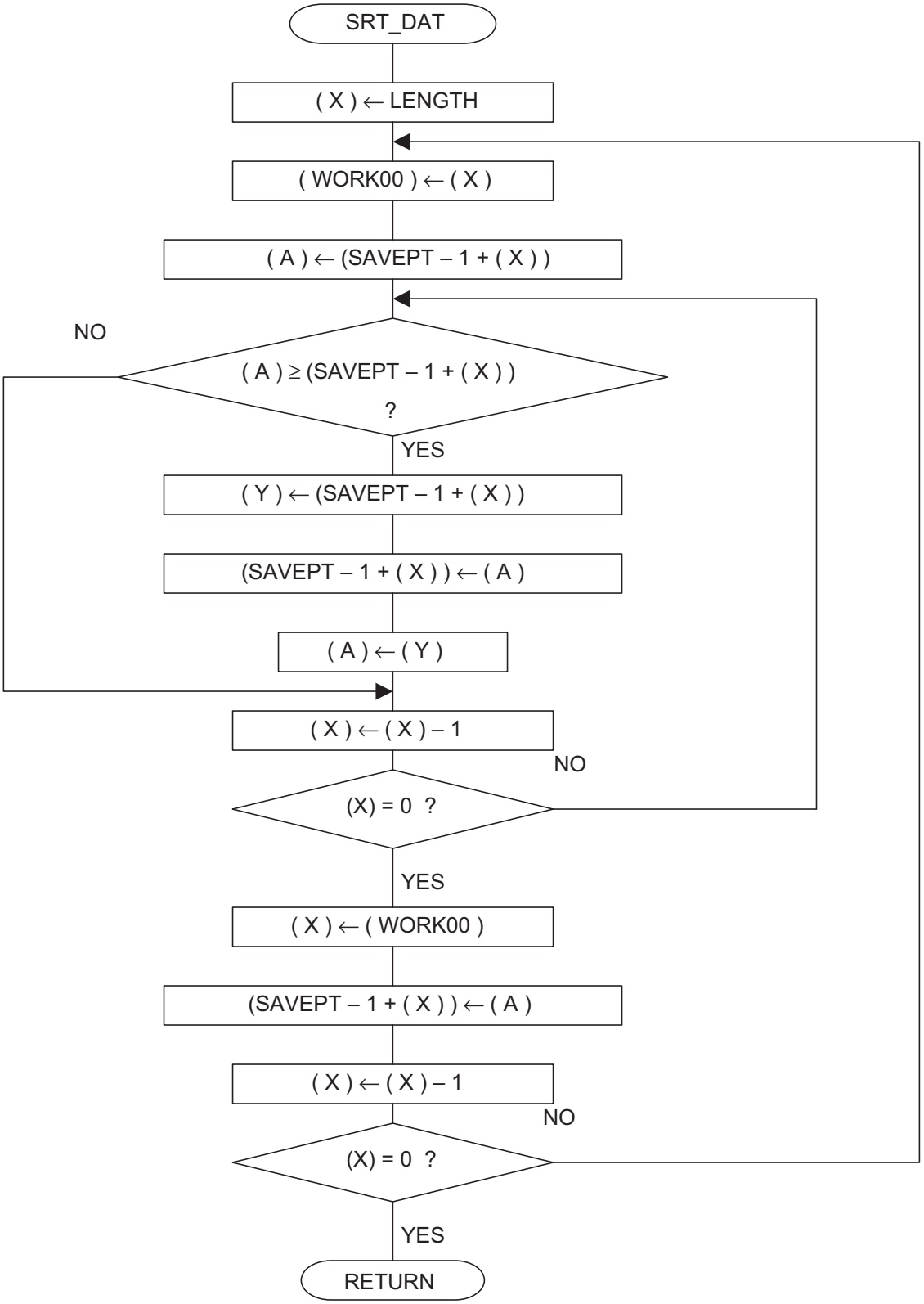
(f) Again, steps (a) through (d) are executed on the sorted area of which the highest address has been decreased, (LENGTH bytes - 2 bytes) from SAVEPT.



(g) In this manner, steps (a) through (d) are executed until the order of the contents is sorted into descending order, decreasing the highest address until it reaches the lowest address.

In addition, in steps (b) through (c), if the contents are switched when the contents of Accumulator A are smaller than those of the memory, the contents can be sorted into ascending order.

(3) Flowchart


SRT_DAT
(X) ← LENGTH
(WORK00) ← (X)
(A) ← (SAVEPT – 1 + (X))
NO
(A) ≥ (SAVEPT – 1 + (X))
?
YES
(Y) ← (SAVEPT – 1 + (X))
(SAVEPT – 1 + (X)) ← (A)
(A) ← (Y)
(X) ← (X) – 1
NO
(X) = 0 ?
YES
(X) ← (WORK00)
(SAVEPT – 1 + (X)) ← (A)
(X) ← (X) – 1
NO
(X) = 0 ?
YES
RETURN

(4) Program List

```
;*******************************************************
;
;               Sort routine
;
;*******************************************************
;
SRT_DAT:
          LDX       #LENGTH           ;Data length
SRT_01:   STX       WORK00
          LDA       SAVEPT-1,X        ;SAVEPT<-->SAVEPT-1+WORK00
SRT_02:   CMP       SAVEPT-1,X        ;If use(BCS)
          BCC       SRT_03            ; -then negative
          LDY       SAVEPT-1,X
          STA       SAVEPT-1,X
          TYA                         ;Minimum data set
SRT_03:   DEX                         ; -to A&Y
          BNE       SRT_02
          LDX       WORK00
          STA       SAVEPT-1,X        ;Minimum data set
          DEX                         ;Next area
          BNE       SRT_01            ;Sort end ?
          RTS                         ;Yes
```

3.6 16-Bit Data Add (Binary)

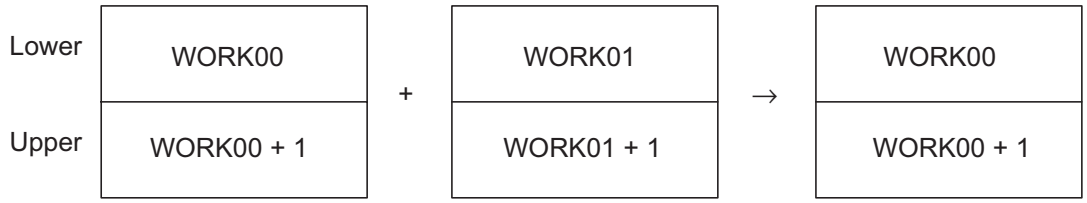
(1) Description

Addition of 16-bit binary data is performed.

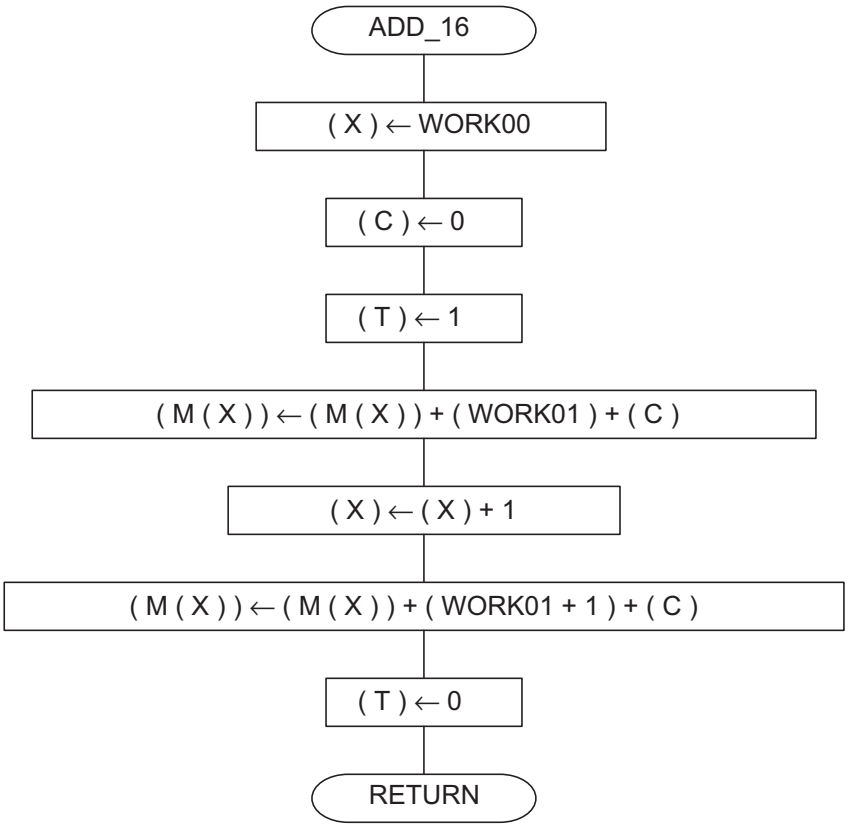
(2) Explanation

The contents of WORK00 +1 and WORK00 are added to the contents of WORK01 +1 and WORK01, respectively; and the results are stored to WORK00 +1 and WORK00, respectively.

When X Modified Operation Mode Flag T is set to “1”, the data is added without destroying the contents of Accumulator A.



(3) Flowchart



Note: M (X) represents the memory of the address indicated by Index Register X.

(4) Program List

```
;*******************************************************
;
;        16 bits BIN. data addition routine
;
;*******************************************************
;
ADD_16:
          LDX       #WORK00
          CLC                           ;C flag clear
          SET                           ;T flag set
          ADC       WORK01
          INX                           ;(WORK00+1)(WORK00)+
          ADC       WORK01+1            ;(WORK01+1)(WORK01)
          CLT                           ;T flag clear
          RTS
```

3.7 16-Bit Data Subtract (Binary)

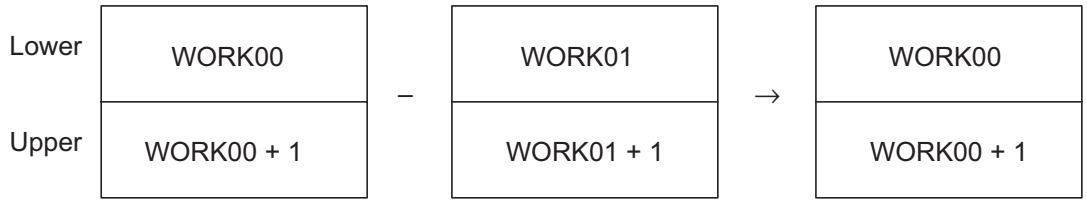
(1) Description

Subtraction of 16-bit binary data is performed.

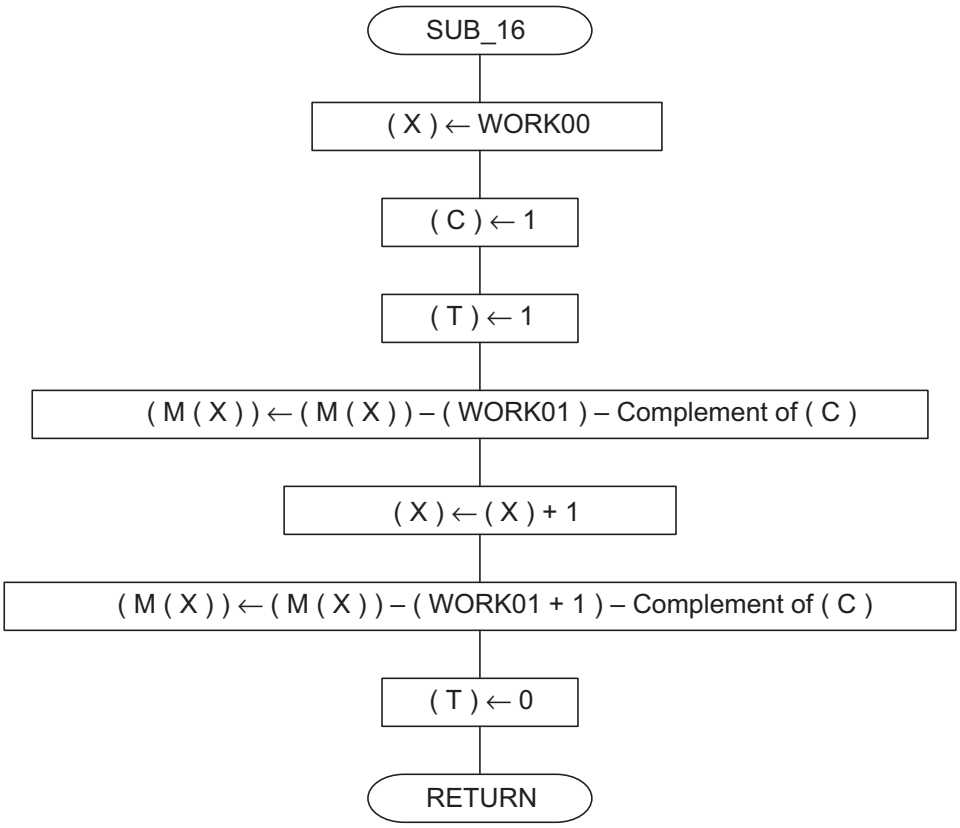
(2) Explanation

The contents of WORK01 +1 and WORK01 are subtracted from the contents of WORK00 +1 and WORK00, respectively, and the results are stored to WORK00 +1 and WORK00, respectively.

When X Modified Operation Mode Flag T is set to “1”, the data is subtracted without destroying the contents of Accumulator A.



(3) Flowchart



Note: M (X) represents the memory of the address indicated by Index Register X.

(4) Program List

```
;*******************************************************
;
;        16 bits BIN. data subtraction routine
;
;*******************************************************
;
SUB_16:
      LDX   #WORK00
      SEC                           ;C flag set
      SET                           ;T flag set
      SBC   WORK01
      INX                           ;(WORK00+1)(WORK00)-SBC
      WORK01+1                      ;(WORK01+1)(WORK01)
      CLT                           ;T flag clear
      RTS
```

3.8 16-Bit Data Add (BCD)

(1) Description

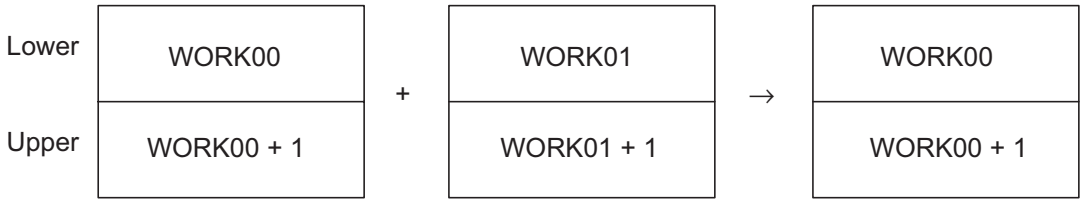
Addition of 16-bit BCD data is performed.

(2) Explanation

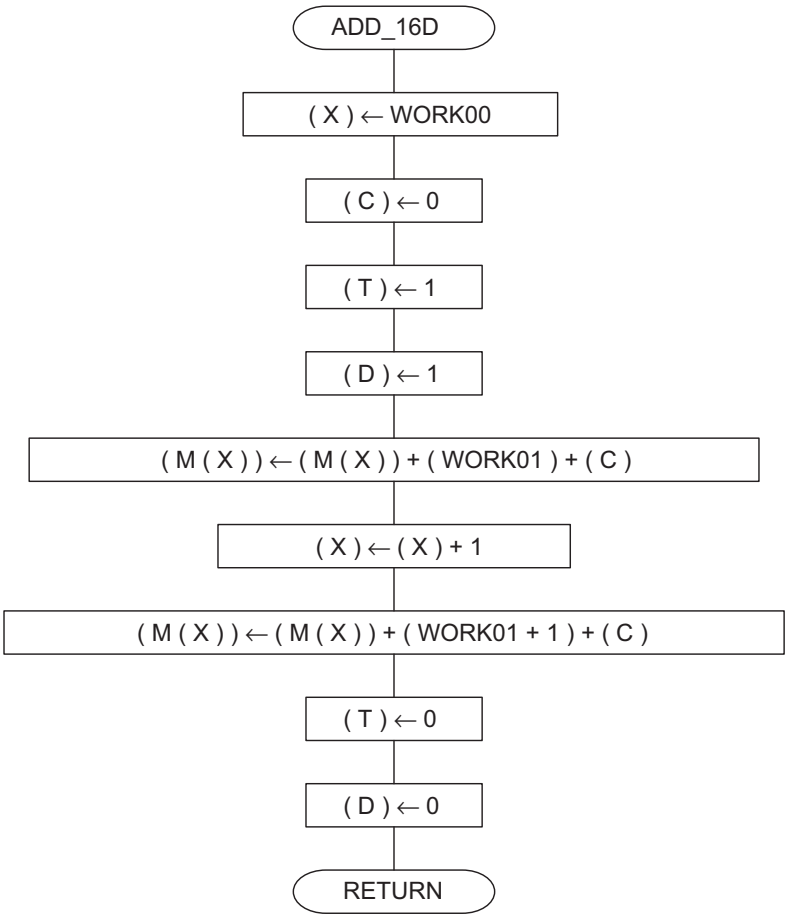
The contents of WORK00 +1 and WORK00 are added to the contents of WORK01 +1 and WORK01, respectively, and the results are stored to WORK00 +1 and WORK00, respectively.

By setting Decimal Mode Flag D to “1”, the ADC instruction can use decimal arithmetic. However, this will delay determination of Carry Flag C, so that make sure the SEC, CLC, and CLD instructions are not executed right after the ADC instruction.

When X Modified Operation Mode Flag T is set to “1”, the data is added without destroying the contents of Accumulator A.



(3) Flowchart



Note: M (X) represents the memory of the address indicated by Index Register X.

(4) Program List

```
;******************************************************
;
;        16 bits BCD data addition routine
;
;******************************************************
ADD_16D:
          LDX       #WORK00
          CLC                         ;C flag reset
          SET                         ;T flag set
          SED                         ;Decimal mode set
          ADC       WORK01
          INX                         ;(WORK00+1)(WORK00)+
          ADC       WORK01+1          ;(WORK01+1)(WORK01)
          CLT                         ;T flag reset
          CLD                         ;Decimal mode clear
          RTS
```

3.9 16-Bit Subtract (BCD)

(1) Description

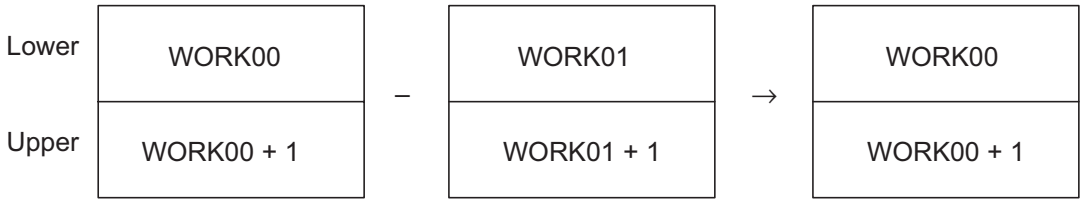
Subtraction of 16-bit BCD data is performed.

(2) Explanation

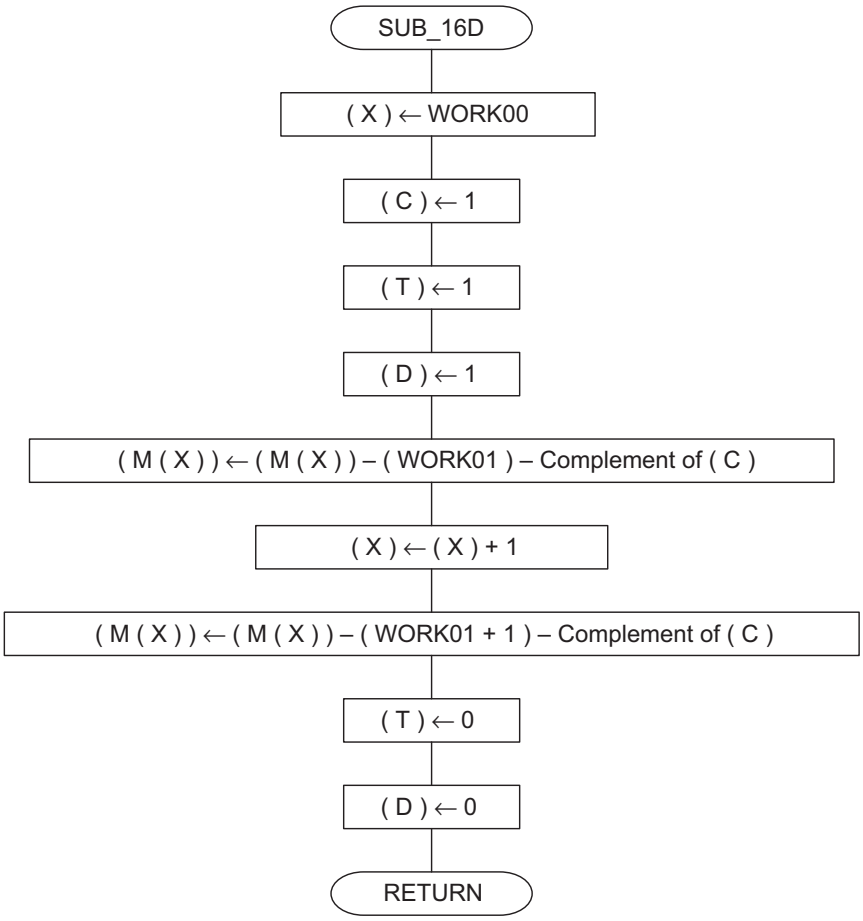
The contents of WORK01 +1 and WORK01 are subtracted from the contents of WORK00 +1 and WORK00, respectively, and the results are stored to WORK00 +1 and WORK00, respectively.

By setting Decimal Mode Flag D to "1", the SBC instruction can use decimal arithmetic. However, this will delay determination of Carry Flag C, so that make sure the SEC, CLC, and CLD instructions are not executed right after the SBC instruction.

When X Modified Operation Mode Flag T is set to "1", the data is subtracted without destroying the contents of Accumulator A.



(3) Flowchart



Note: M (X) represents the memory of the address indicated by Index Register X.

(4)Program List

```
;******************************************************
;
;        16 bits BCD data subtraction routine
;
;******************************************************
SUB_16D:
          LDX       #WORK00
          SEC                         ;C flag set
          SET                         ;T flag set
          SED                         ;Decimal mode set
          SBC       WORK01
          INX                         ;(WORK00+1)(WORK00)-
          SBC       WORK01+1          ;(WORK01+1)(WORK01)
          CLT                         ;T flag reset
          CLD                         ;Decimal mode reset
          RTS
```

3.10 16-Bit Data Multiply (Binary)

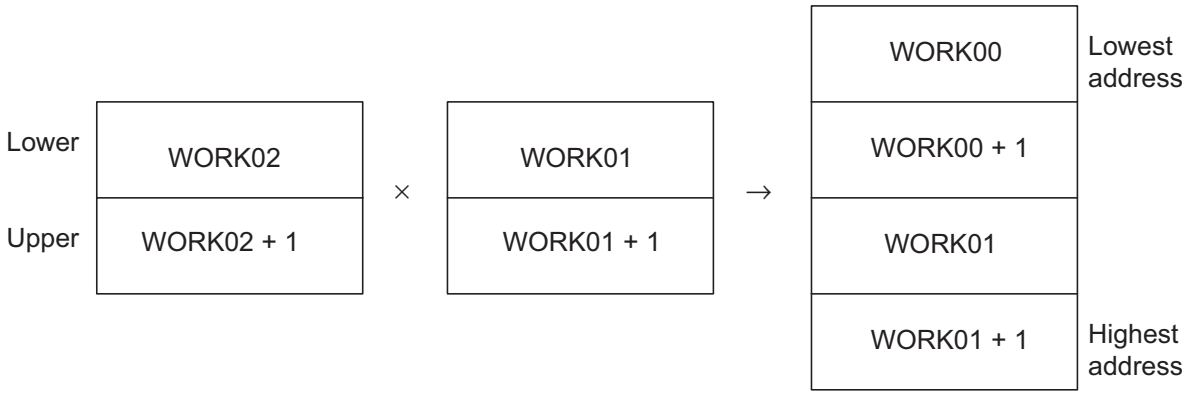
(1) Description

Multiplication of 16-bit binary data is performed.

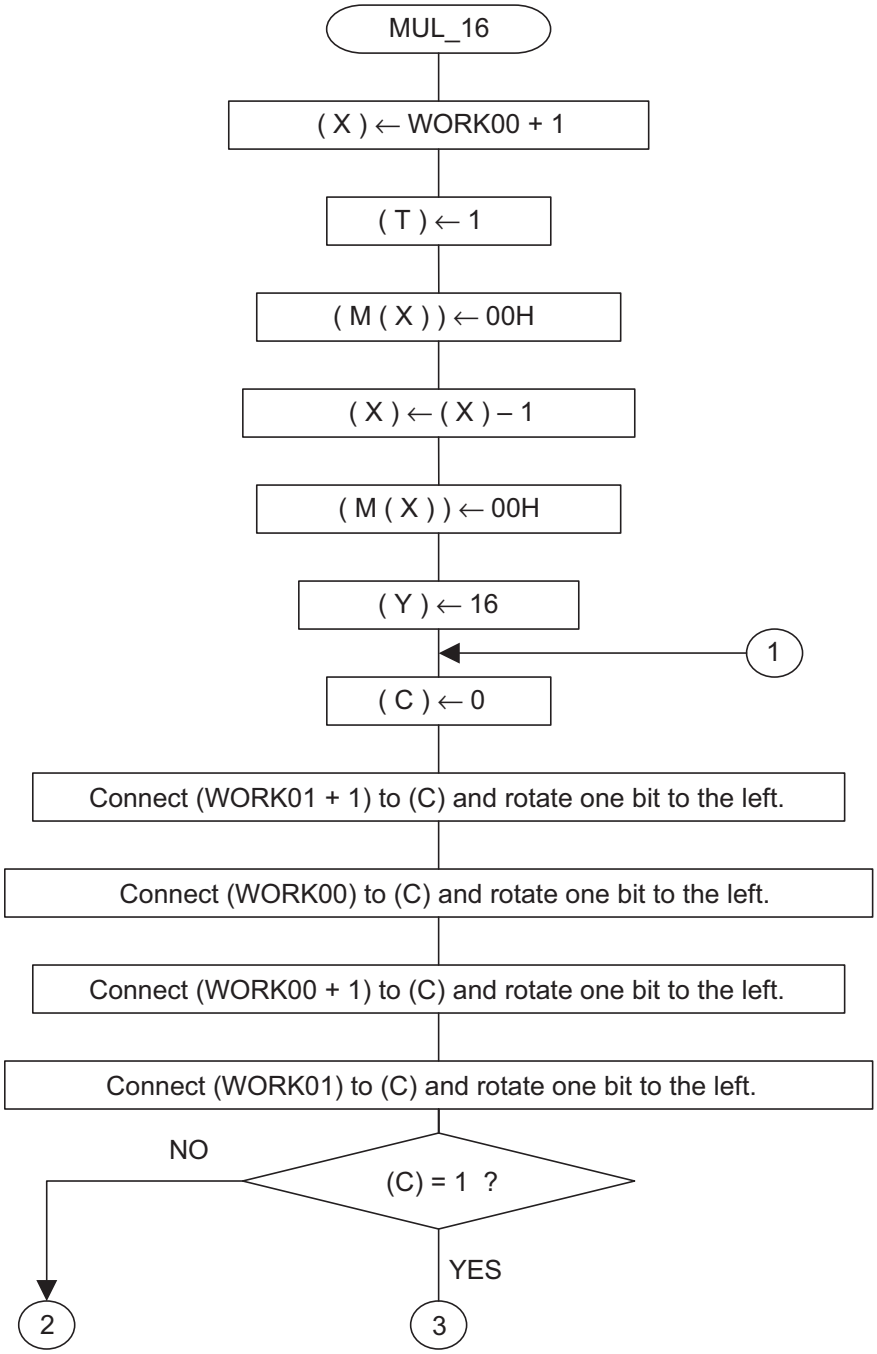
(2) Explanation

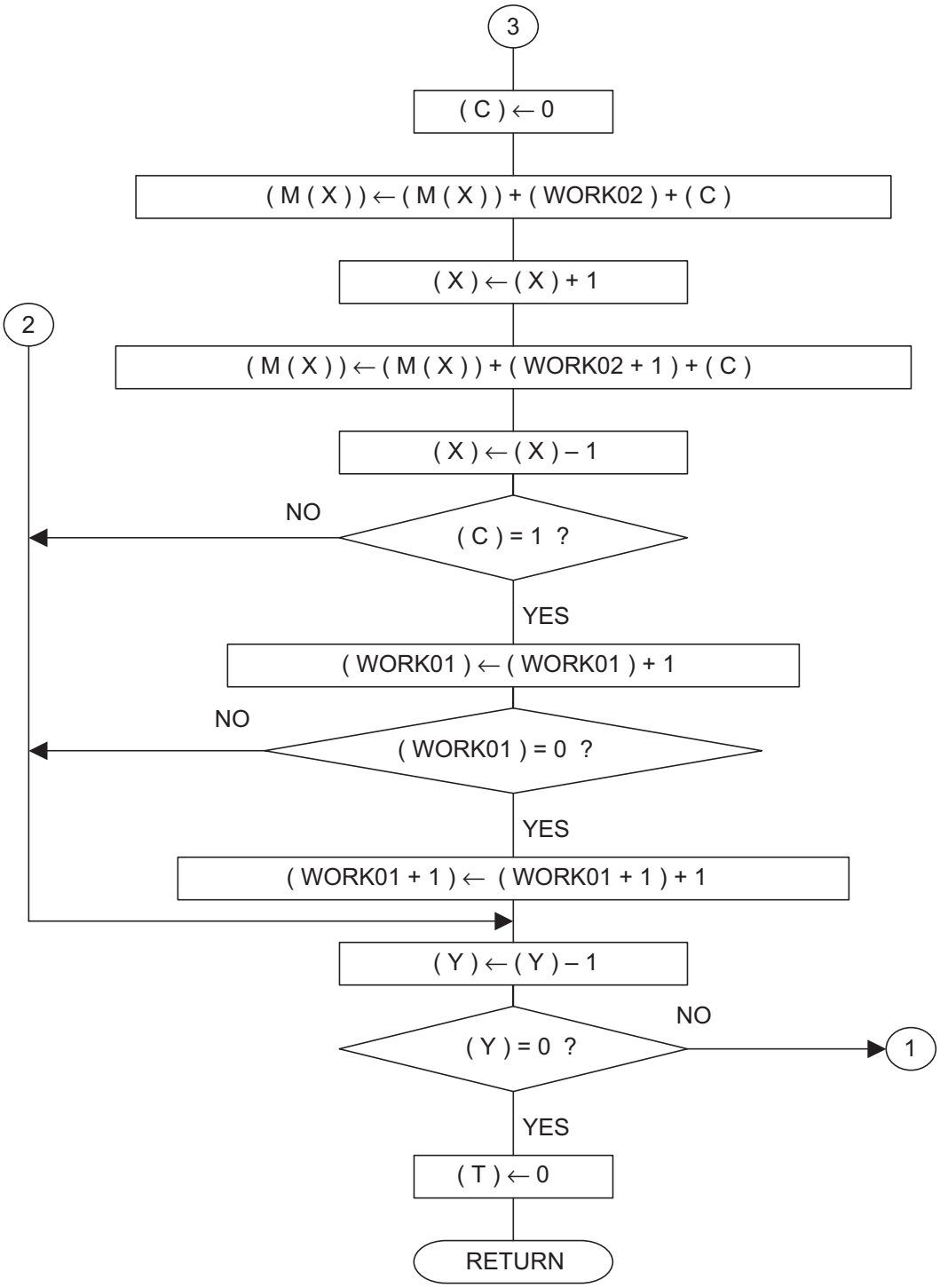
The contents of WORK02 +1 and WORK02 are multiplied by the contents of WORK01 +1 and WORK01, respectively, and the results are stored to WORK01 +1, WORK01, WORK00 +1 and WORK00.

When X Modified Operation Mode Flag T is set to “1”, the data is multiplied without destroying the contents of Accumulator A.



(3) Flowchart


MUL_16
(X) ← WORK00 + 1
(T) ← 1
(M (X)) ← 00H
(X) ← (X) – 1
(M (X)) ← 00H
(Y) ← 16
1
(C) ← 0
Connect (WORK01 + 1) to (C) and rotate one bit to the left.
Connect (WORK00) to (C) and rotate one bit to the left.
Connect (WORK00 + 1) to (C) and rotate one bit to the left.
Connect (WORK01) to (C) and rotate one bit to the left.
NO
(C) = 1 ?
YES
2
3



Note: M (X) represents the memory of the address indicated by Index Register X.

(4) Program List

```
;******************************************************
;
;        16 bits BIN. data multiplication routine
;
;******************************************************
;
MUL_16:
          LDX      #WORK00+1           ;Product L addr. set
          SET                          ;T flag set
          LDA      #$00                ;Clear product L
          DEX
          LDA      #$00
          LDY      #16                 ;Bit counter set
;------------------------------
MUL_01:   CLC
          ROL      WORK00              ;Rotate product L
          ROL      WORK00+1
          ROL      WORK01              ;Rotate product H
          ROL      WORK01+1
          BCC      MUL_02              ;C flag 1 ?
          CLC                          ;Yes
          ADC      WORK02              ;Multiplicand + product L
          INX
          ADC      WORK02+1
          DEX
          BCC      MUL_02              ;Over flow ?
          INC      WORK01              ;Yes
          BNE      MUL_02              ;Over flow ?
          INC      WORK01+1            ;Yes
;------------------------------
MUL_02:   DEY
          BNE      MUL_01              ;Multiple end ?
          CLT                          ;Yes
          RTS
```

3.11 16-Bit Data Divide (Binary)

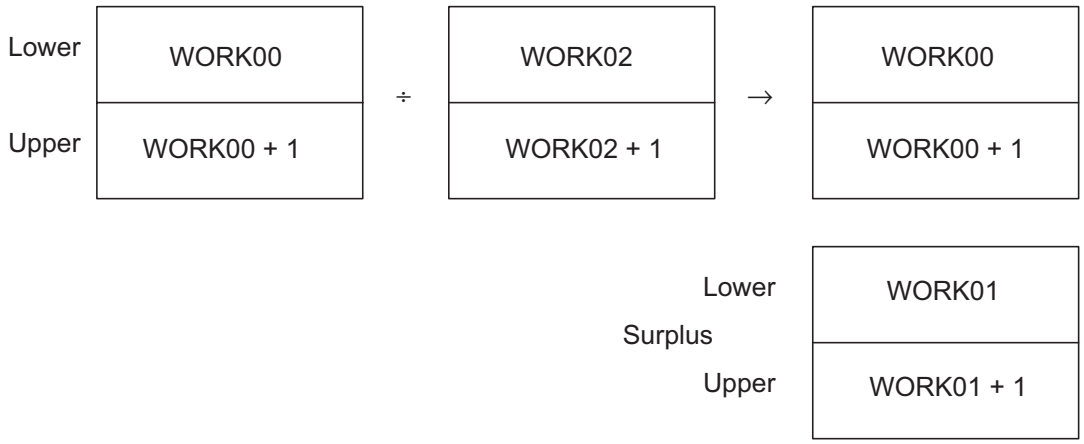
(1) Description

Division of 16-bit binary data is performed.

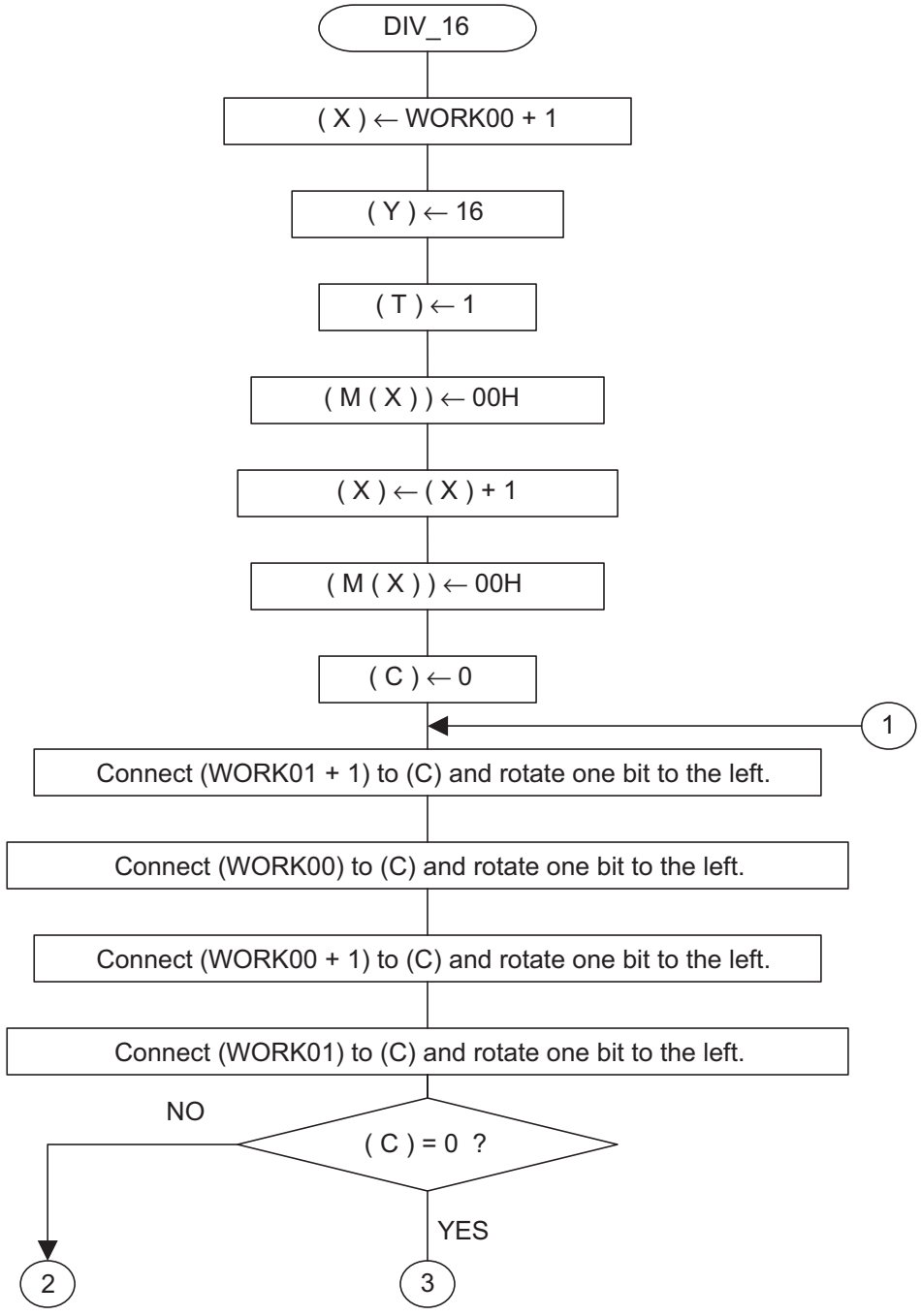
(2) Explanation

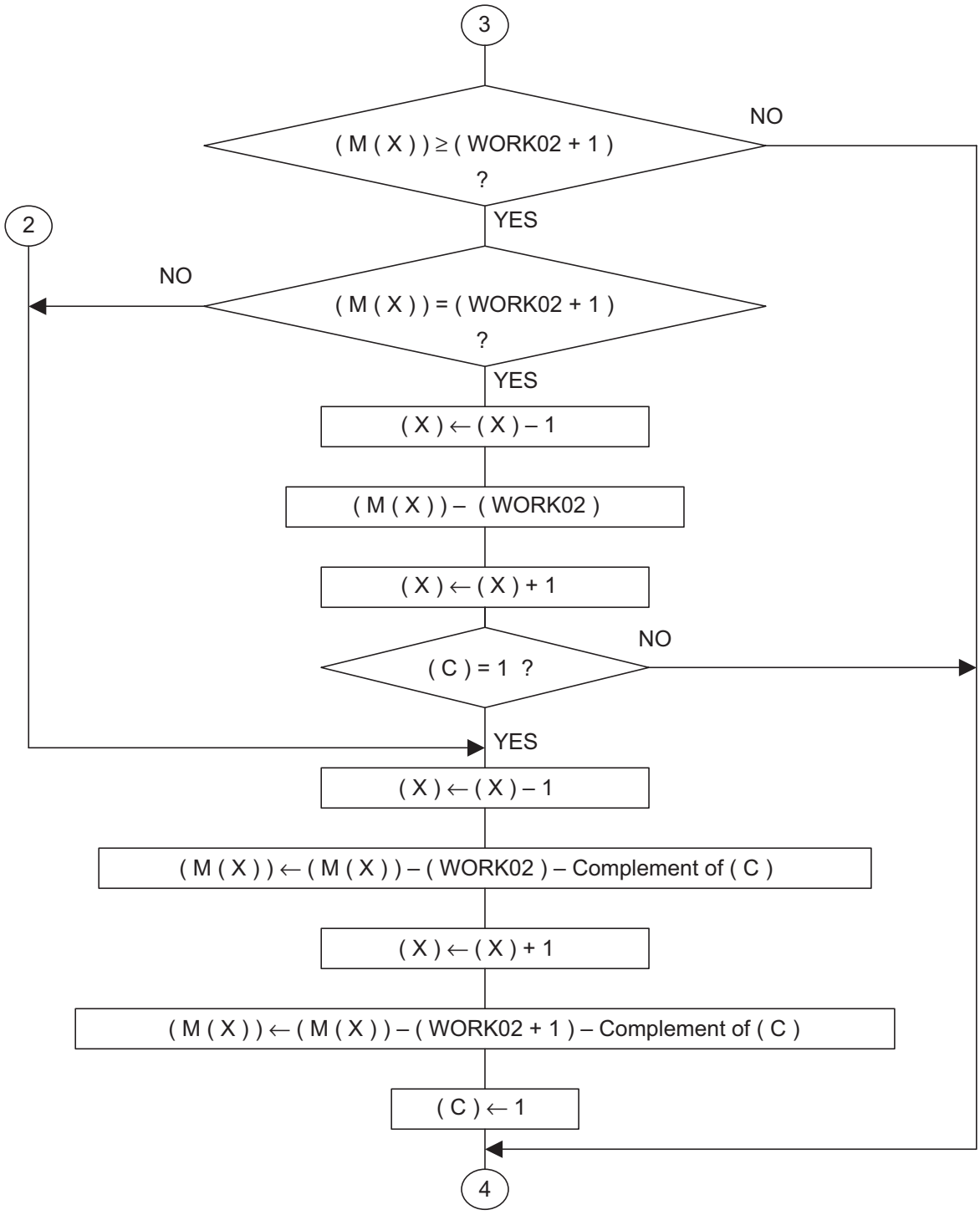
The contents of WORK00 +1 and WORK00 are divided by the contents of WORK02 +1 and WORK02. The quotients are stored to WORK00 +1, and WORK00; and the surpluses are stored to WORK01 +1 and WORK01.

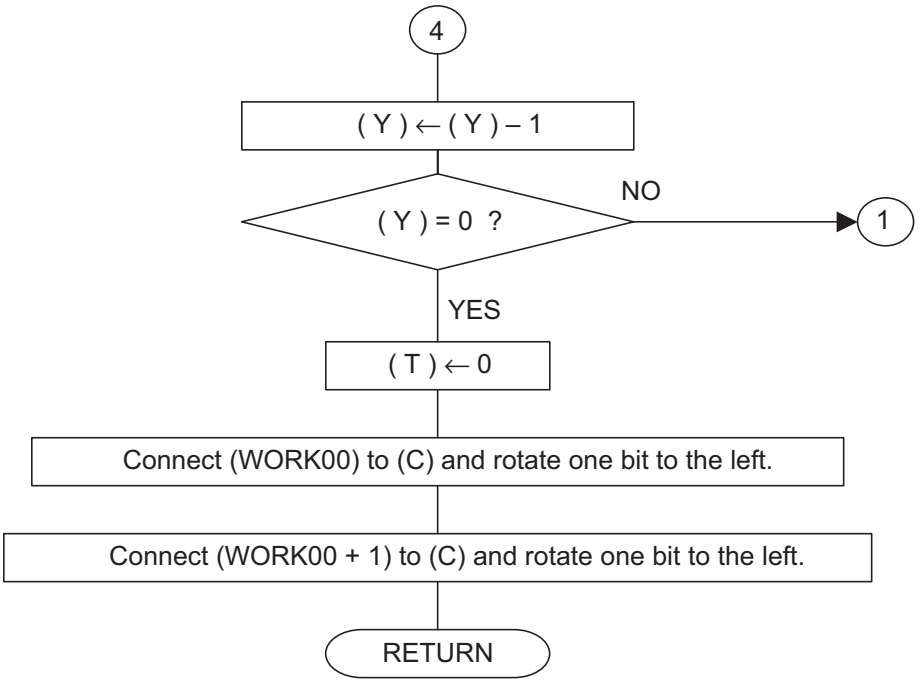
When X Modified Operation Mode Flag T is set to “1”, the data is divided without destroying the contents of Accumulator A.



(3) Flowchart


DIV_16
(X) ← WORK00 + 1
(Y) ← 16
(T) ← 1
(M (X)) ← 00H
(X) ← (X) + 1
(M (X)) ← 00H
(C) ← 0
1
Connect (WORK01 + 1) to (C) and rotate one bit to the left.
Connect (WORK00) to (C) and rotate one bit to the left.
Connect (WORK00 + 1) to (C) and rotate one bit to the left.
Connect (WORK01) to (C) and rotate one bit to the left.
NO
(C) = 0 ?
YES
2
3


3
(M (X)) ≥ (WORK02 + 1)
?
NO
YES
2
NO
(M (X)) = (WORK02 + 1)
?
YES
(X) ← (X) – 1
(M (X)) – (WORK02)
(X) ← (X) + 1
(C) = 1 ?
NO
YES
(X) ← (X) – 1
(M (X)) ← (M (X)) – (WORK02) – Complement of (C)
(X) ← (X) + 1
(M (X)) ← (M (X)) – (WORK02 + 1) – Complement of (C)
(C) ← 1
4



Note: M (X) represents the memory of the address indicated by Index Register X.

(4) Program List

```
;******************************************************
;
;       16 bits BIN. data division routine
;
;******************************************************
;
DIV_16:
         LDX     #WORK01             ;Surplus addr. set
         LDY     #16                 ;Bit counter set
         SET                         ;T flag set
         LDA     #$00                ;Clear surplus
         INX
         LDA     #$00
         CLC
DIV_01:  ROL     WORK00              ;Rotate quotient
         ROL     WORK00+1
         ROL     WORK01              ;Rotate surplus
         ROL     WORK01+1
         BCS     DIV_02              ;C flag 1 ?
;--------------------------------
         CMP     WORK02+1            ;No
         BCC     DIV_03              ;Cannot divide ?
         BNE     DIV_02              ;No
         DEX
         CMP     WORK02
         INX
         BCC     DIV_03              ;Cannot divide ?
;--------------------------------
DIV_02:  DEX                         ;No
         SBC     WORK02              ;Surplus - divisor
         INX
         SBC     WORK02+1
         SEC
DIV_03:  DEY
         BNE     DIV_01              ;Divide end ?
;--------------------------------
         CLT                         ;Yes
         ROL     WORK00              ;Rotate quotient
         ROL     WORK00+1
         RTS
```

4. Application Program Example

4.1 File Handling (transfer)

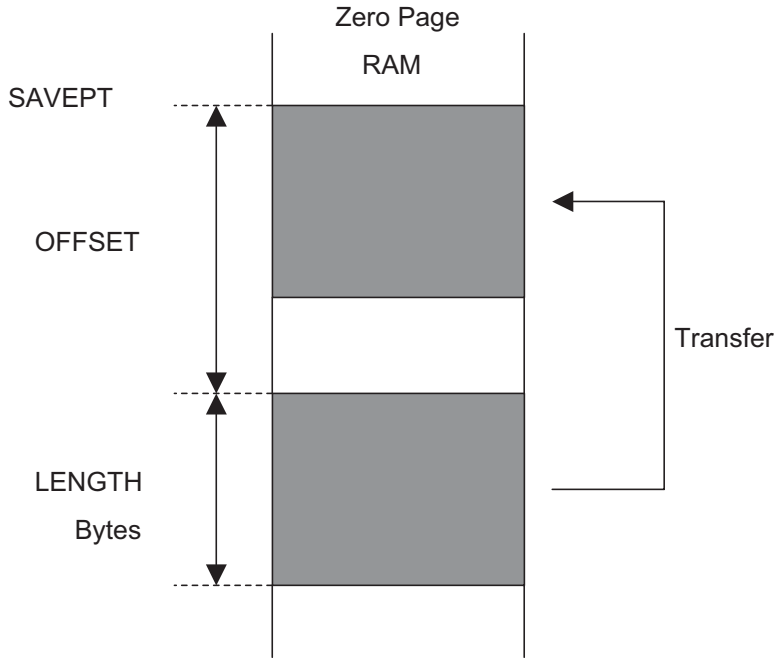
(1) Description

This operation recognizes a part of the zero page RAM as the file memory and executes the data transfer process from one location to another.

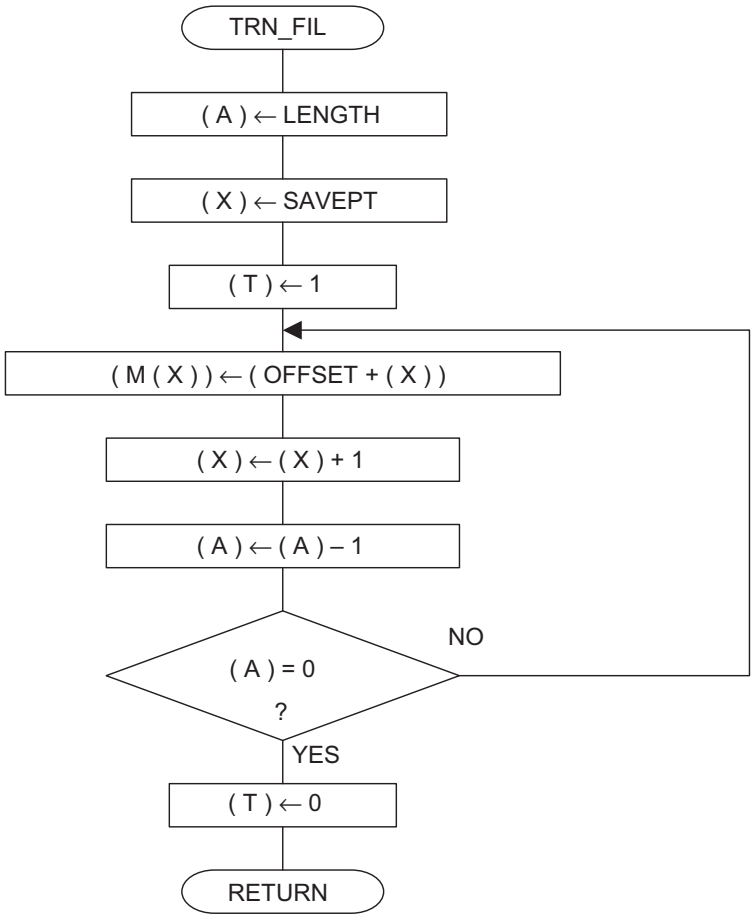
(2) Explanation

LENGTH bytes of file memory data are transferred from the zero page RAM address SAVEPT + OFFSET to continuous zero page RAM addresses starting at SAVEPT.

X Modified Operation Mode Flag T is set, and then the data is transferred between memories. The pointer is set to single pointer + OFFSET, not double pointer (source/target).



(3) Flowchart



Note: M (X) represents the memory of the address indicated by Index Register X.

(4) Program List

```
;******************************************************
;
;           File handling (transfer)
;
;******************************************************
TRN_FIL:
          LDA       #LENGTH          ;File length
          LDX       #SAVEPT
          SET                        ;T flag set
TRN_01:
          LDA       OFFSET,X         ;Transfer data from
          INX                        ; -SAVEPT+OFFSET
          DEC       A                ; -to SAVEPT
          BNE       TRN_01           ;Transfer end ?
          CLT                        ;Yes
          RTS
```

4.2 File Handling (exchange)

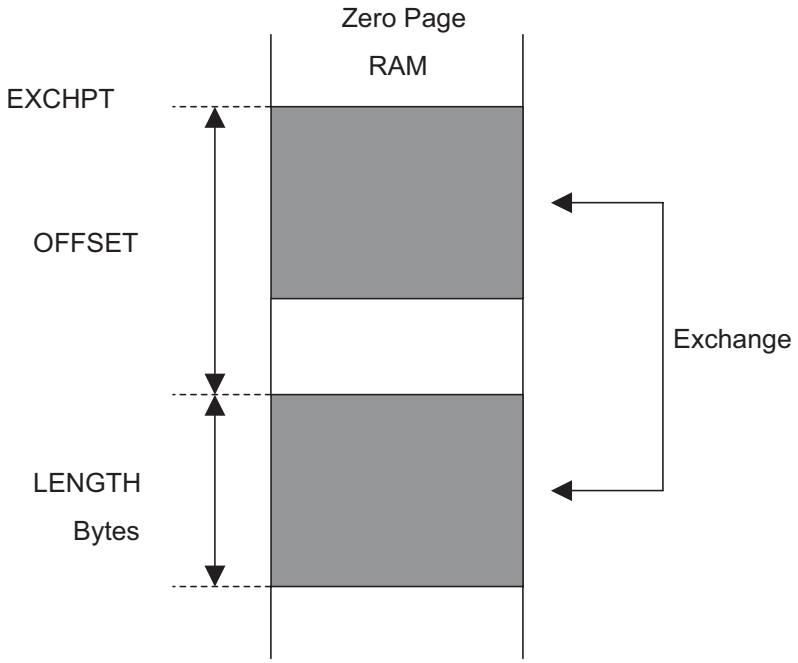
(1) Description

This operation recognizes a part of the zero page RAM as the file memory and executes the data exchange process from one location to another.

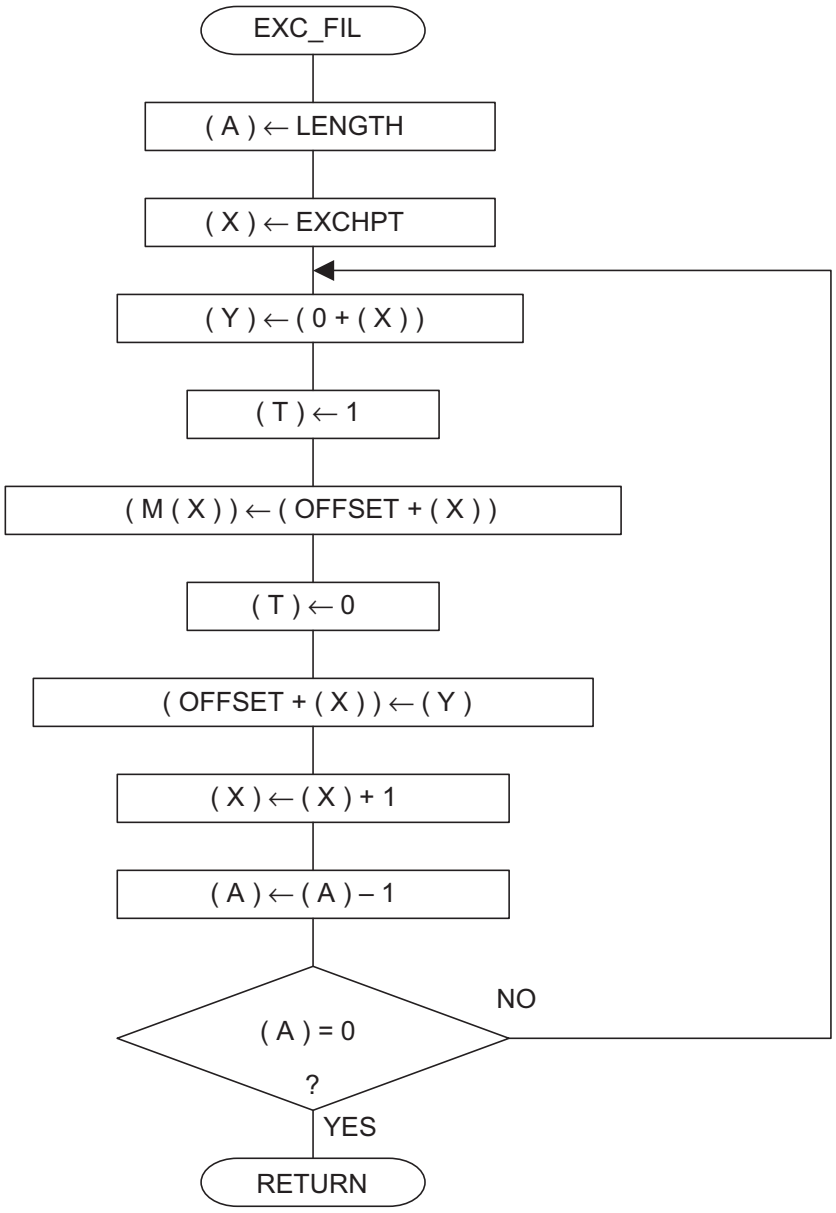
(2) Explanation

LENGTH bytes of file memory data from the zero page RAM address EXCHPT + OFFSET are exchanged with the data from zero page RAM address EXCHPT.

X Modified Operation Mode Flag T is set, and then the data is transferred between memories. The pointer is set to single pointer + OFFSET, not double pointer (source/target).



(3) Flowchart



Note: M (X) represents the memory of the address indicated by Index Register X.

(4) Program List

```
;*******************************************************
;
;           File handling (exchange)
;
;*******************************************************
;
EXC_FIL:
          LDA       #LENGTH             ;File length
          LDX       #EXCHPT
EXC_01:
          LDY       0,X                 ;Exchange data of
          SET                           ; -EXCHPT+OFFSET
          LDA       OFFSET,X            ; -with EXCHPT
          CLT
          STY       OFFSET,X
          INX
          DEC       A
          BNE       EXC_01              ;Exchange end ?
          RTS                           ;Yes
```

4.3 Code Conversion (packed BCD → unpacked BCD)

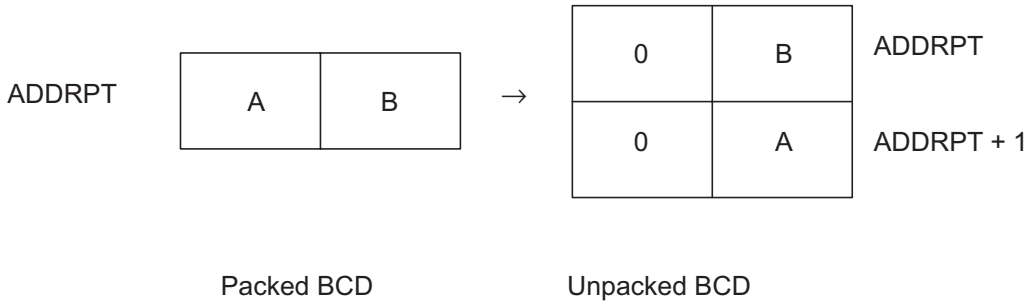
(1) Description

Packed BCD data is converted to unpacked BCD data.

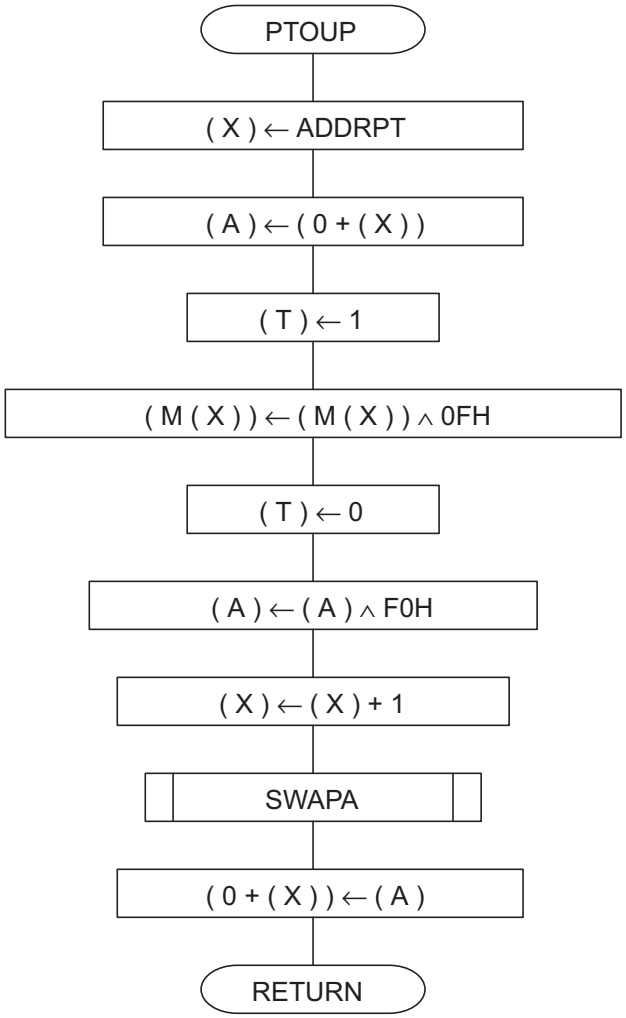
(2) Explanation

The packed BCD data at zero page RAM address ADDRPT is converted to the unpacked BCD data and stored at ADDRPT and ADDRPT + 1.

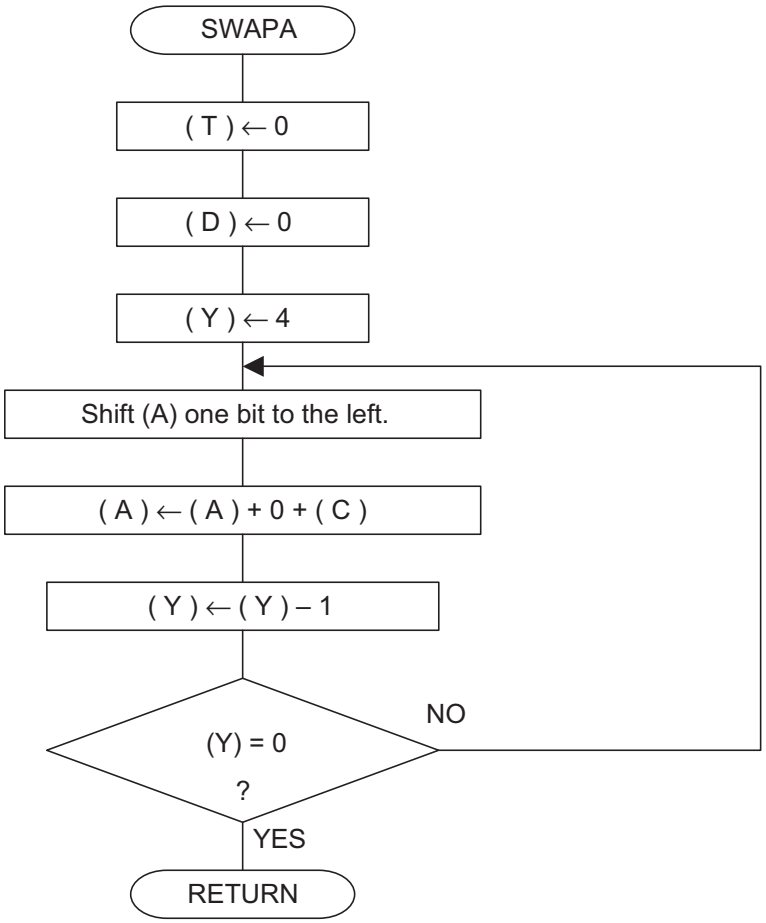
A packed BCD consists of two decimal digits in one byte. But packed-to-unpacked conversion results in two unpacked BCD bytes. Each unpacked BCD byte has one digit in the lower four bits and zero filling the upper four bits.



(3) Flowchart



Note: M (X) represents the memory of the address indicated by Index Register X.


SWAPA
(T) ← 0
(D) ← 0
(Y) ← 4
Shift (A) one bit to the left.
(A) ← (A) + 0 + (C)
(Y) ← (Y) – 1
(Y) = 0
?
NO
YES
RETURN

(4) Program List

```
;******************************************************
;
;            Packed BCD -> unpacked BCD
;
;******************************************************
;
PTOUP:
          LDX       #ADDRPT
          LDA       0,X             ;Get packed BCD data
          SET                       ;T flag set
          AND       #0FH            ;Unpacked BCD data L
          CLT                       ;T flag clear
          AND       #0F0H
          INX
          JSR       SWAPA           ;Swap A
          STA       0,X             ;Unpacked BCD data H
          RTS
;******************************************************
;
;            Swap A register
;
;******************************************************
;
SWAPA:
          CLT                       ;T flag clear
          CLD                       ;Decimal mode clear
          LDY       #4
SWAPA1:
          ASL       A
          ADC       #0
          DEY
          BNE       SWAPA1
          RTS
```

4.4 Code Conversion (unpacked BCD → packed BCD)

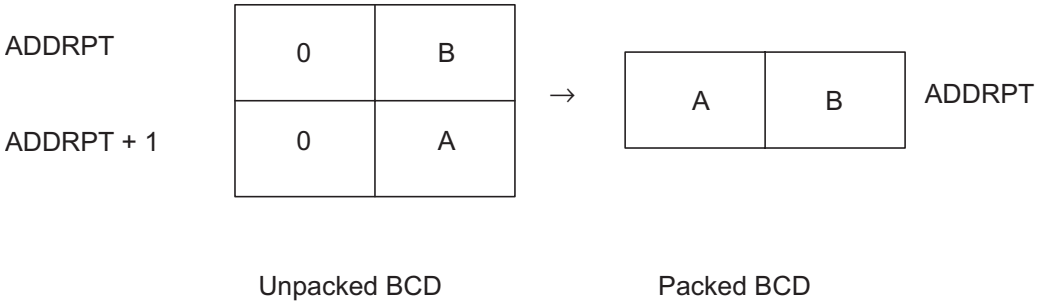
(1) Description

Unpacked BCD data is converted to packed BCD data.

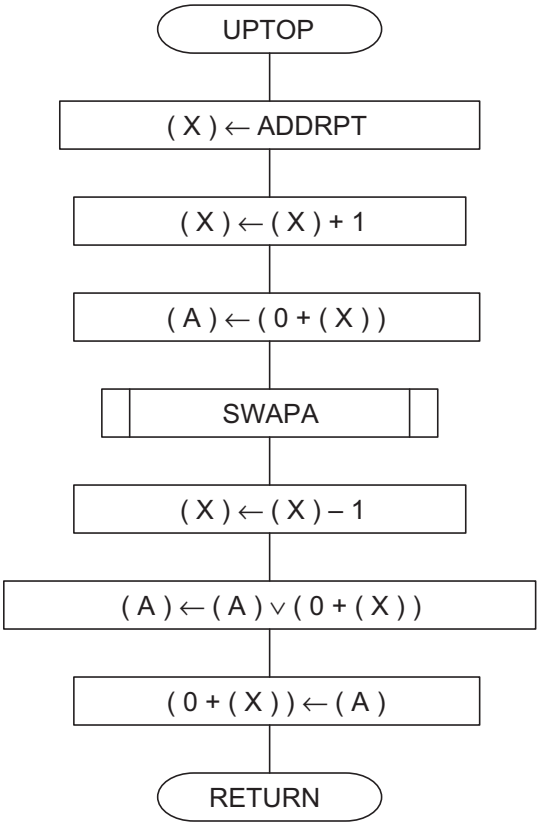
(2) Explanation

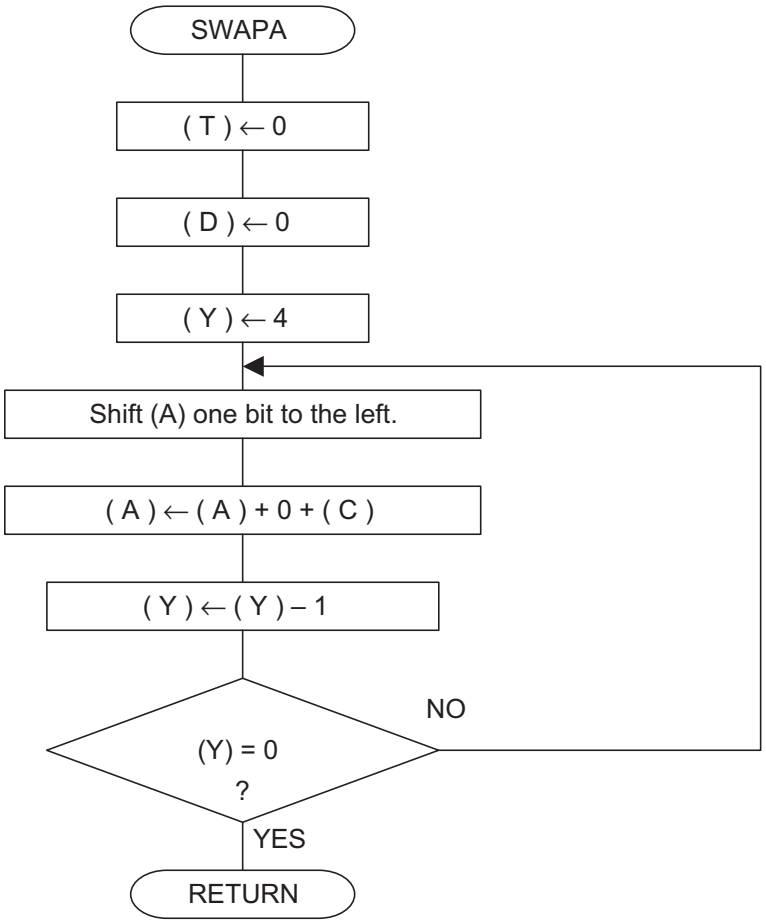
The unpacked BCD data at zero page RAM address ADDRPT +1 and ADDRPT are converted to the packed BCD data and stored at ADDRPT.

A packed BCD consists of two decimal digits in one byte. But packed-to-unpacked conversion results in two unpacked BCD bytes. Each unpacked BCD byte has one digit in the lower four bits and zero filling the upper four bits.



(3) Flowchart




SWAPA
(T) ← 0
(D) ← 0
(Y) ← 4
Shift (A) one bit to the left.
(A) ← (A) + 0 + (C)
(Y) ← (Y) – 1
(Y) = 0
?
NO
YES
RETURN

(4) Program List

```
;******************************************************
;
;            Unpacked BCD -> packed BCD
;
;******************************************************
;
UPTOP:
          LDX       #ADDRPT
          INX
          LDA       0,X                 ;Get unpacked BCD data H
          JSR       SWAPA               ;Swap A
          DEX
          ORA       0,X                 ;Packed BCD data
          STA       0,X
          RTS
;******************************************************
;
;            Swap A register
;
;******************************************************
;
SWAPA:
          CLT                           ;T flag clear
          CLD                           ;Decimal mode clear
          LDY       #4
SWAPA1:
          ASL       A
          ADC       #0
          DEY
          BNE       SWAPA1
          RTS
```

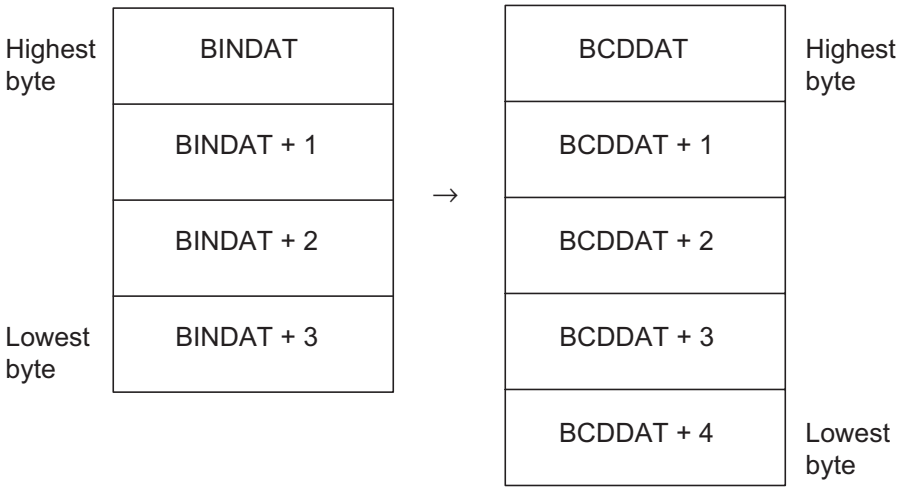
4.5 Code Conversion (BIN → BCD)

(1) Description

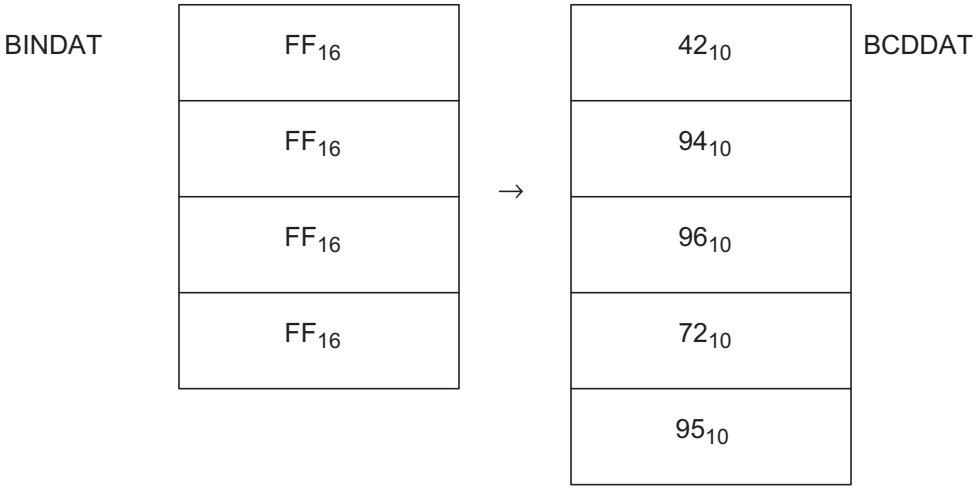
4-byte BIN data is converted to 5-byte BCD data.

(2) Explanation

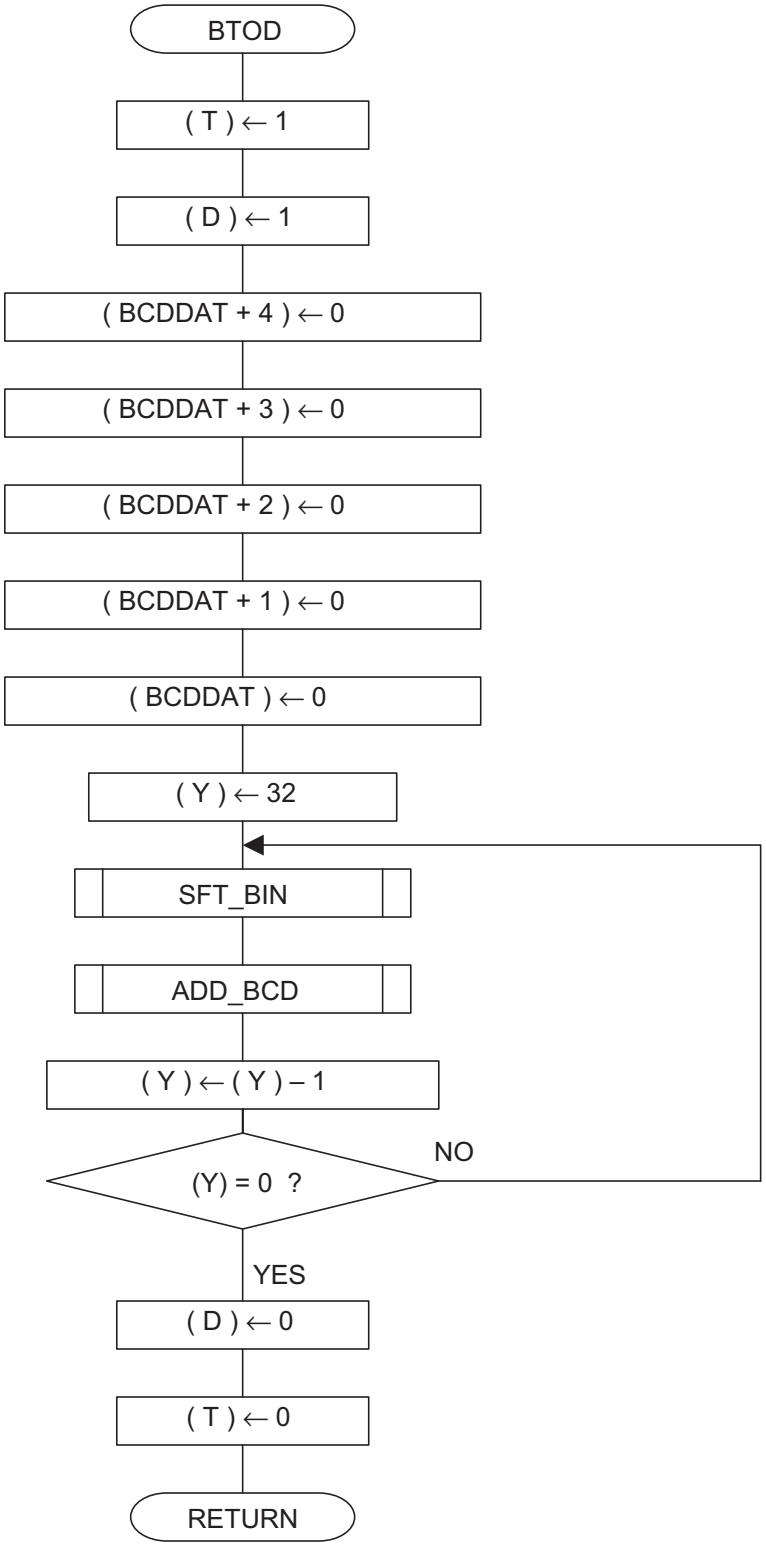
The 4-byte BIN data at zero page RAM address BINDAT, BINDAT +1, BINDAT +2, and BINDAT +3 are converted to 5-byte BCD data and stored at BCDDAT, BCDDAT +1, BCDDAT +2, BCDDAT +3 and BCDDAT +4, respectively.

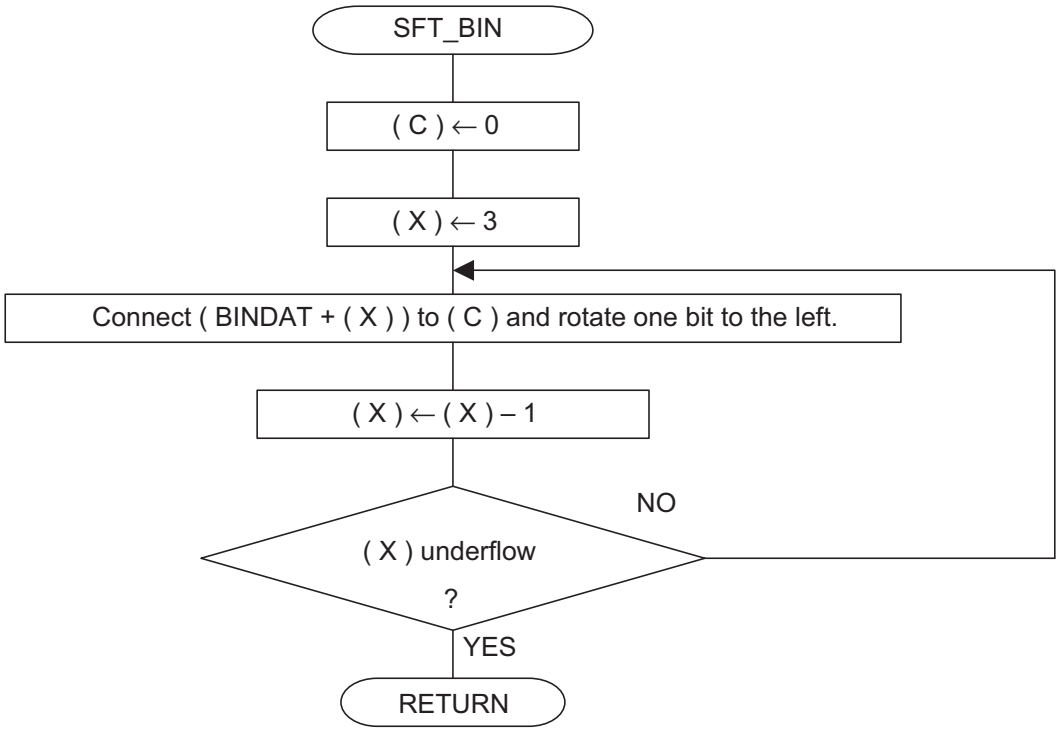


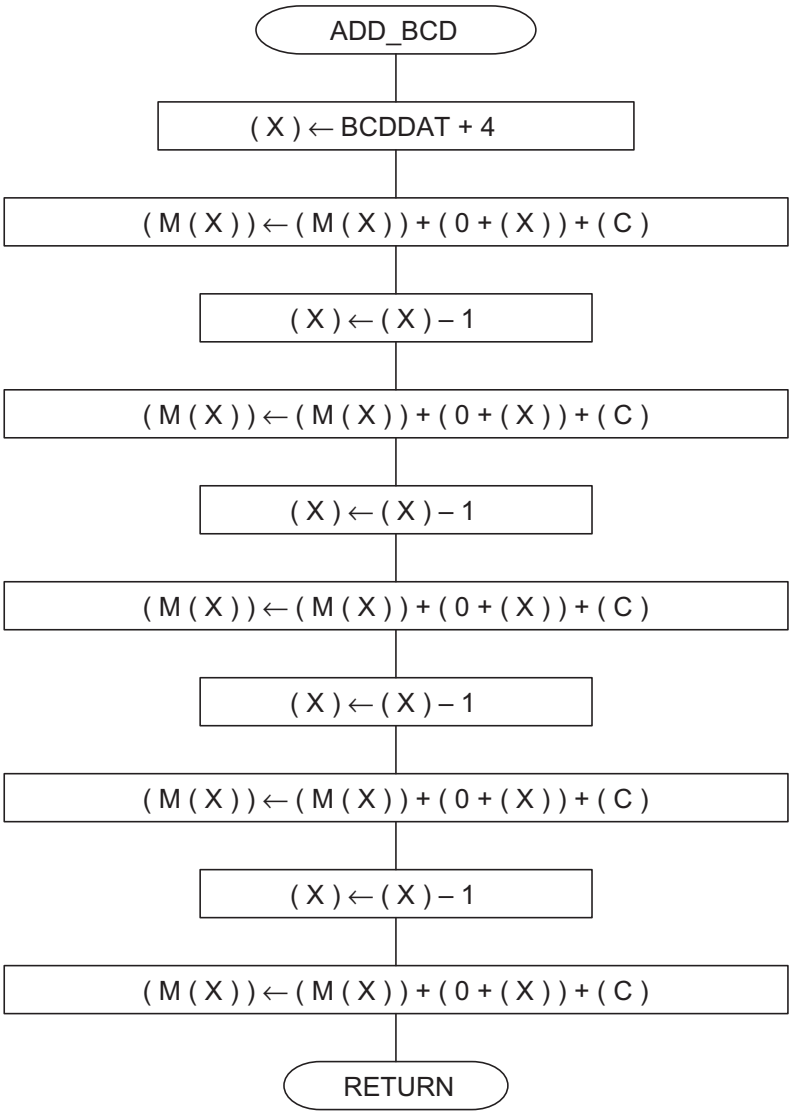
Example: If the BIN data is FFFFFFFFH, the BCD data will be 4294967295.



(3) Flowchart


BTOD
(T) ← 1
(D) ← 1
(BCDDAT + 4) ← 0
(BCDDAT + 3) ← 0
(BCDDAT + 2) ← 0
(BCDDAT + 1) ← 0
(BCDDAT) ← 0
(Y) ← 32
SFT_BIN
ADD_BCD
(Y) ← (Y) – 1
(Y) = 0 ?
NO
YES
(D) ← 0
(T) ← 0
RETURN


SFT_BIN
(C) ← 0
(X) ← 3
Connect (BINDAT + (X)) to (C) and rotate one bit to the left.
(X) ← (X) – 1
NO
(X) underflow
?
YES
RETURN



Note: M (X) represents the memory of the address indicated by Index Register X.

(4) Program List

```
;******************************************************
;
;               BIN -> BCD
;
;******************************************************
BTOD:
          SET                               ;T flag set
          SED                               ;Decimal mode set
          LDM       #0,BCDDAT+4             ;Clear BCD result area
          LDM       #0,BCDDAT+3
          LDM       #0,BCDDAT+2
          LDM       #0,BCDDAT+1
          LDM       #0,BCDDAT
          LDY       #32                     ;Yes
BTOD_01:
          JSR       SFT_BIN                 ;Left shift BIN data
          JSR       ADD_BCD                 ;2*(BCD)+C -> (BCD)
          DEY
          BNE       BTOD_01                 ;Convert end ?
          CLD                               ;Yes
          CLT                               ;T flag clear
          RTS
;******************************************************
;
;            Left shift BIN data
;
;******************************************************
SFT_BIN:
          CLC                             ;C flag clear
          LDX       #3
SFT_01:
          ROL       BINDAT,X
          DEX
          BPL       SFT_01                ;Shift end ?
          RTS                             ;Yes
;******************************************************
;
;            2*(BCD)+C -> (BCD)
;
;******************************************************
ADD_BCD:
          LDX       #BCDDAT+4
          ADC       0,X
          DEX
          ADC       0,X
          DEX
          ADC       0,X
          DEX
          ADC       0,X
          DEX
          ADC       0,X
          RTS
```

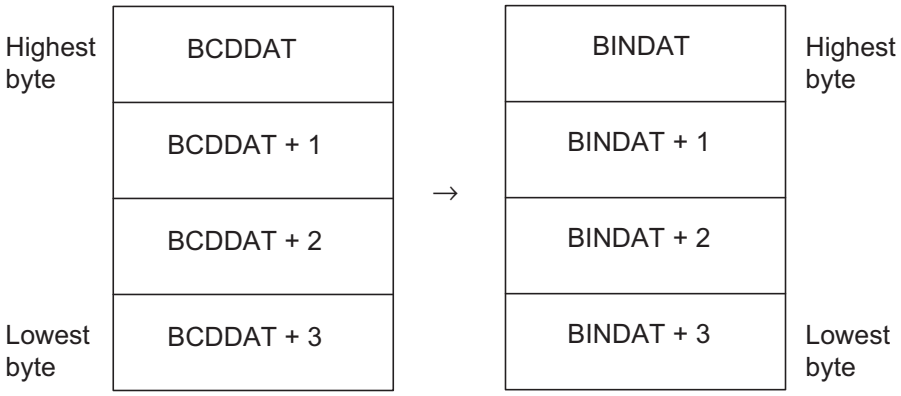
4.6 Code Conversion (BCD → BIN)

(1) Description

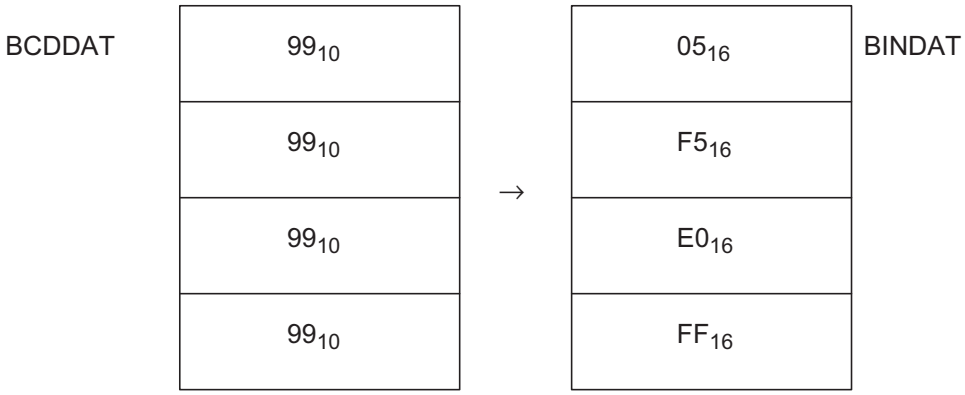
4-byte BCD data is converted to 4-byte BIN data.

(2) Explanation

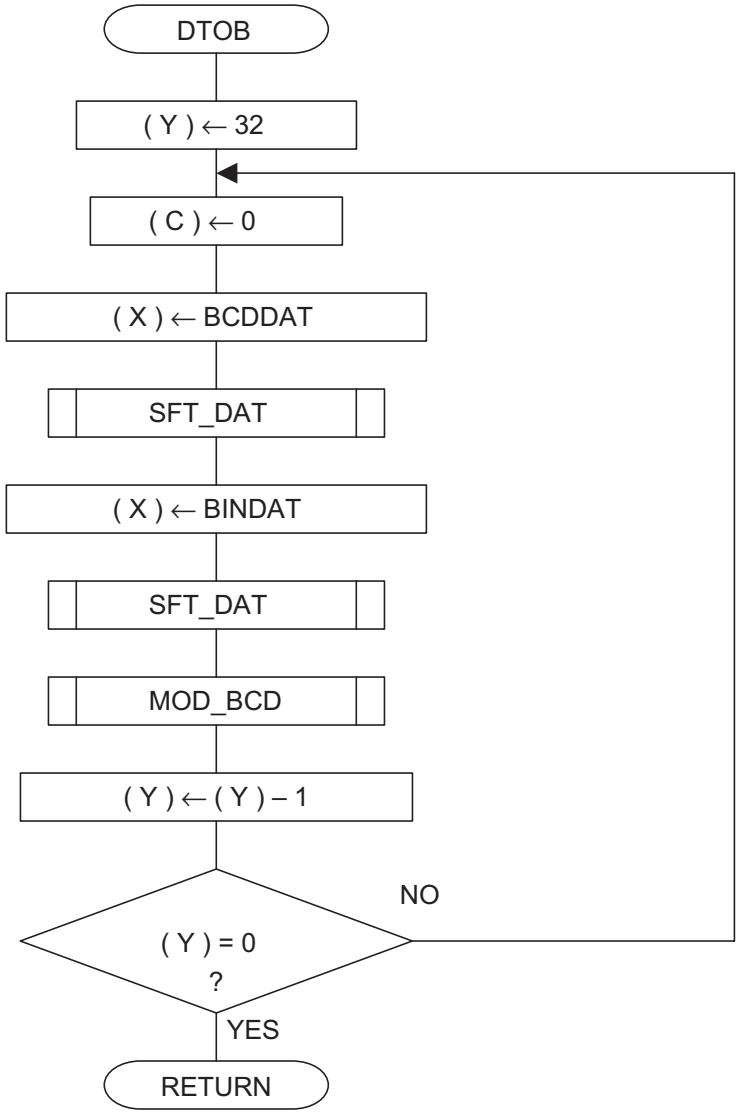
The 4-byte BCD data at zero page RAM addresses BCDDAT, BCDDAT +1, BCDDAT +2, and BCDDAT +3 are converted to 4-byte BIN data and stored at BINDAT, BINDAT +1, BINDAT +2, and BINDAT +3, respectively.

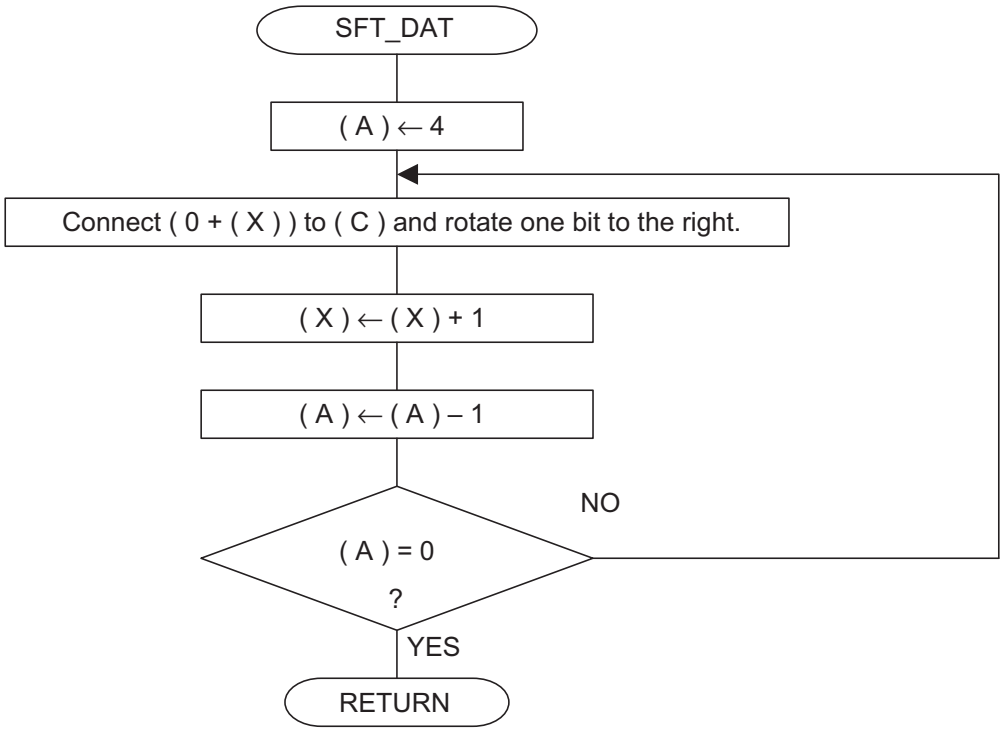


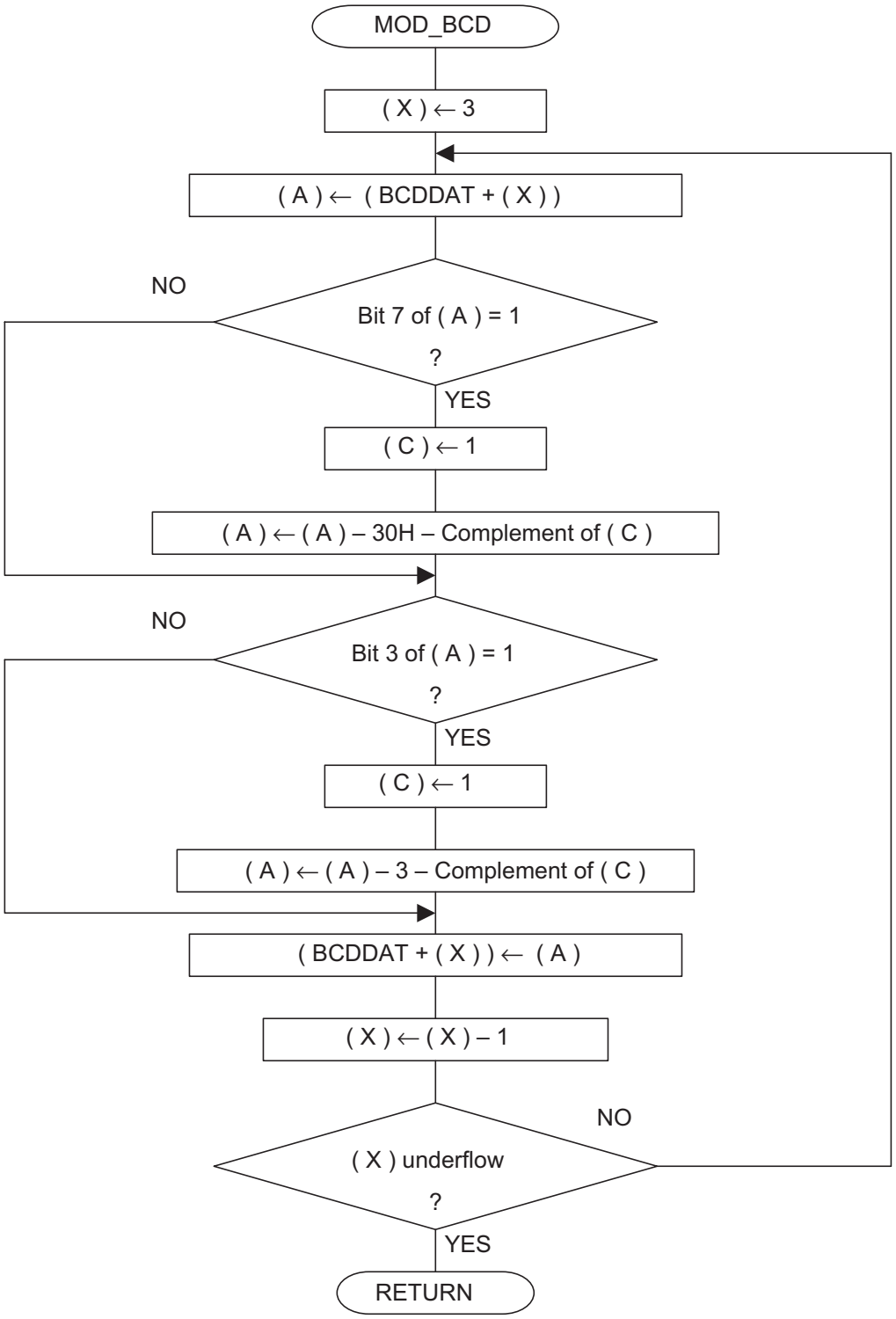
Example: If the BCD data is 99999999, the BIN data will be 05F5E0FFH.



(3) Flowchart


DTOB
(Y) ← 32
(C) ← 0
(X) ← BCDDAT
SFT_DAT
(X) ← BINDAT
SFT_DAT
MOD_BCD
(Y) ← (Y) – 1
(Y) = 0
?
NO
YES
RETURN


SFT_DAT
(A) ← 4
Connect (0 + (X)) to (C) and rotate one bit to the right.
(X) ← (X) + 1
(A) ← (A) – 1
NO
(A) = 0
?
YES
RETURN


MOD_BCD
(X) ← 3
(A) ← (BCDDAT + (X))
NO
Bit 7 of (A) = 1
?
YES
(C) ← 1
(A) ← (A) – 30H – Complement of (C)
NO
Bit 3 of (A) = 1
?
YES
(C) ← 1
(A) ← (A) – 3 – Complement of (C)
(BCDDAT + (X)) ← (A)
(X) ← (X) – 1
NO
(X) underflow
?
YES
RETURN

(4) Program List

```
;******************************************************
;
;           BCD -> BIN
;
;******************************************************
DTOB:
          LDY       #32
DT0B_01:
          CLC
          LDX       #BCDDAT             ;Point to BCDDAT
          JSR       SFT_DAT             ;Right shift BCD data
          LDX       #BINDAT             ;Point to BINDAT
          JSR       SFT_DAT             ;Right shift BIN data
          JSR       MOD_BCD             ;Get modified BCD data
          DEY
          BNE       DT0B_01             ;Shift end ?
          RTS       ;Yes !
;******************************************************
;
;           Right shift data
;
;******************************************************
SFT_DAT:
          LDA       #4
SFT_02:
          ROR       0,X
          INX
          DEC       A
          BNE       SFT_02
          RTS
;******************************************************
;
;           Modify BCD data
;
;******************************************************
MOD_BCD:
          LDX   #3
MOD_01:
          LDA       BCDDAT,X
          BBC       7,A,MOD_02
          SEC
          SBC       #30H
MOD_02:
          BBC       3,A,MOD_03
          SEC
          SBC       #3
MOD_03:
          STA       BCDDAT,X
          DEX
          BPL       MOD_01
          RTS
```

4.7 SGN Function

(1) Description

This is the SGN function for Accumulator A.

(2) Explanation

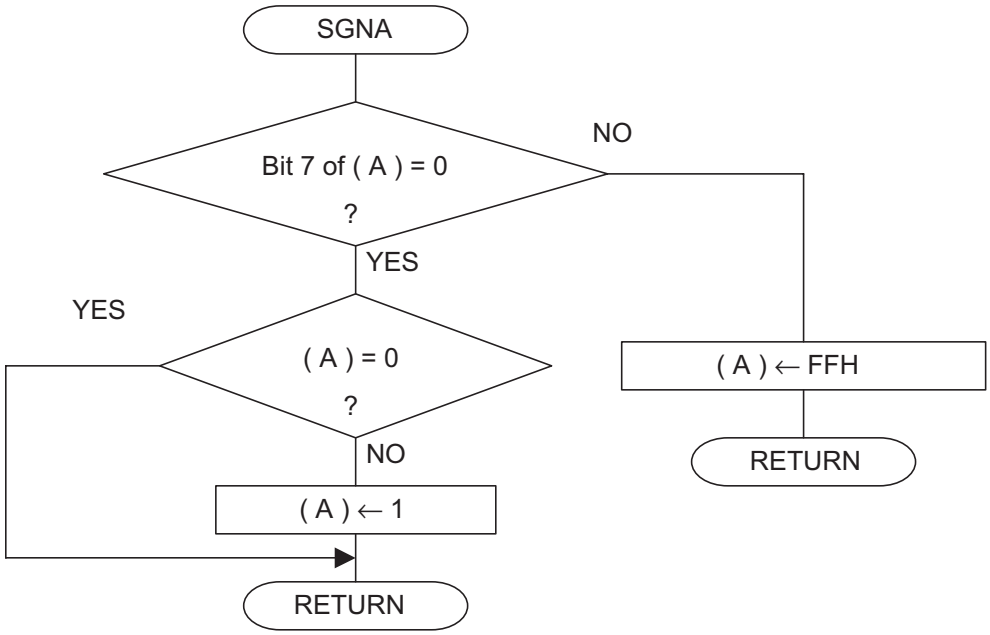
The SGN value for the contents of Accumulator A is figured and stored into Accumulator A. Bit 7 of Accumulator A is a sign bit. The SGN function for this operation is as follows:

When (A) > 0, SGN (A) = 1

When (A) = 0, SGN (A) = 0

When (A) < 0, SGN (A) = –1

(3) Flowchart



(4) Program List

```
;*******************************************************
;
;                       SGN(A)
;
;*******************************************************
;
SGNA:
          BBS       7,A,SGN_01
          CMP       #0
          BEW       SGN_02
          LDA       #1
SGN_02:
          RTS
SGN_01:
          LDA       #$FF
          RTS
```

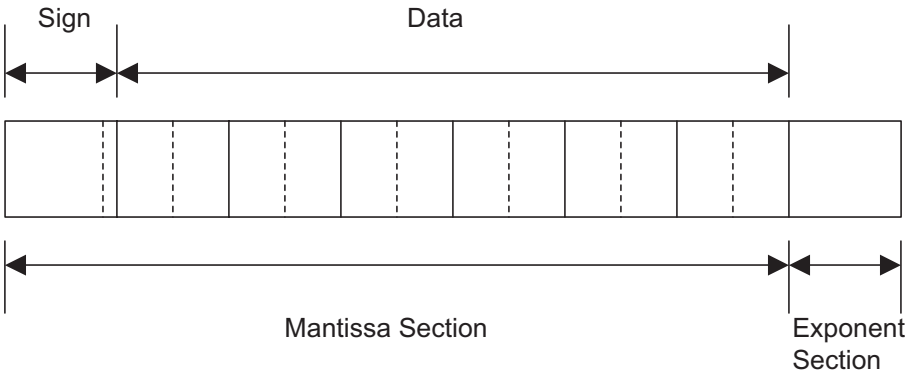
4.8 BCD 12-digit Floating Point Arithmetic Calculations

(1) Description

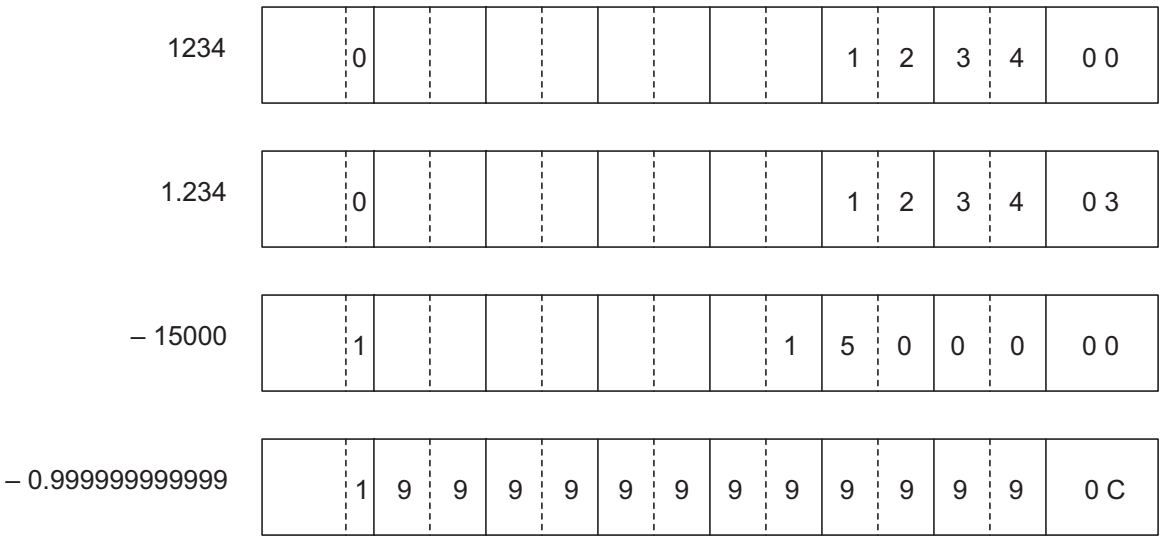
Arithmetic calculations for BCD 12-digit floating point numbers are performed.

(2) Explanation

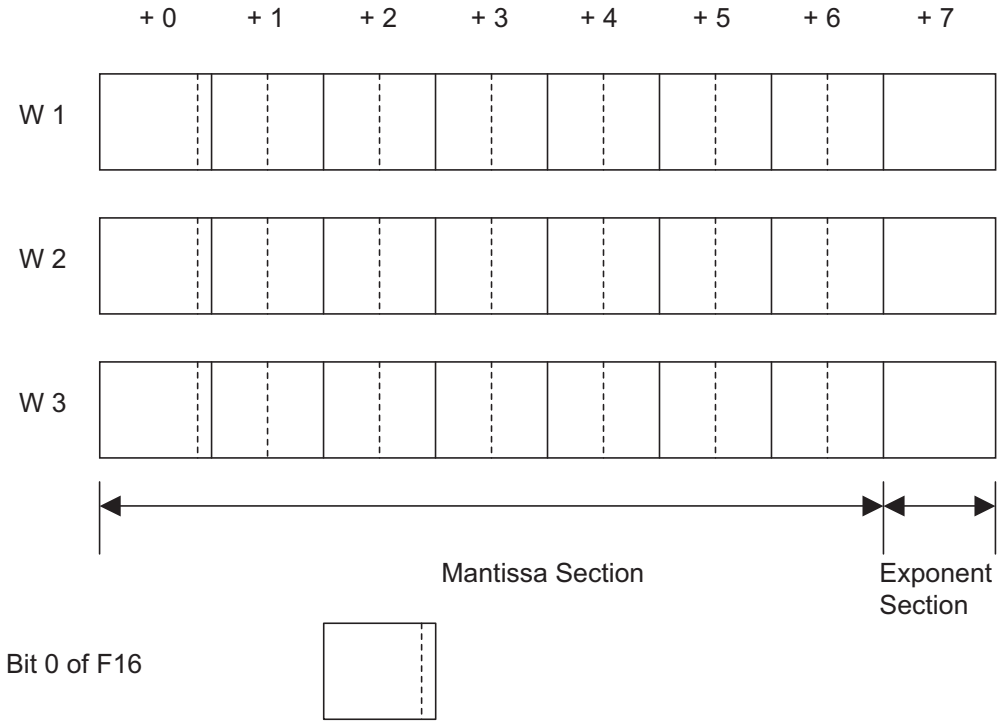
The data format, as shown below, consists of two sections: the mantissa (7 bytes: 1 byte of sign bit, 6 bytes of data) and the exponent (one byte). "0" indicates a positive mantissa sign; "1" indicates a negative mantissa sign. The mantissa section data consists of 12 digits of valid numerical numbers (BCD). The exponent section can only handle values ranging from 00H to 0CH; the results of the calculation cannot be guaranteed for values ranging from 0DH to FFH.



The following are examples of numbers expressed in this format.



This format is used by the following three files in the zero page RAM area: W1, W2, and W3.



- W1: Stores the number to be operated and the results.
- W2: Stores the operand. Address is W2 = W1 + 8.
- W3: Uses as a work file for multiply and divide operations. Address is W3 = W2 + 8.
- Bit 0 of F16: Goes to "1" when error occurs.

The following shows the calling sequence for each arithmetic operation. Note that add and subtract are determined by the value in bit 3 of W2.

- Multiply:
  (a) Set the number to be operated in W1.
  (b) Set the operand in W2.
  (c) JSR MULT

- Divide:
  Steps (a) & (b) are the same as those in the Multiply operation.
  (c) JSR DIV
- Add:

Steps (a) & (b) are the same as those in the Multiply operation.

(c) JSR ADSB

- Subtract:
  Steps (a) & (b) are the same as those in the Multiply operation.
  (c) Set bit 3 of W2 to "1".
  (d) JSR ADSB

After these processes are executed, the results can be checked for errors with the data in bit 0 of zero page RAM address F16. If bit 0 of F16 = "1", an error has occurred. Errors that may occur in these operations are described below.

Operation	Error
Add	When the result exceeds 999999999999
Subtract	When the result exceeds – 999999999999
Multiply	When the result exceeds ± 999999999999
Divide	• When it is divided by 0 • When the result exceeds ± 999999999999

(3) Program List

```
;*******************************************************
;
;       12 digits BCD number addition,subtraction
;       ,multiplication,division
;
;*******************************************************
;
;
;       RAM ASSIGN
;
;    0   1    2     3     4     5     6    7
;W1:SIGN MSD<----------------------------LSD INDX
;W2:SIGN MSD<----------------------------LSD INDX
;W3:SIGN MSD<----------------------------LSD INDX
;
;
;*******************************************************
;
;       W1 <- W1/W2
;
;*******************************************************
;
DIVI:
        JSR     BFITL                   ;W3<-W1 copy,W1<-0
        SBC     8,X                     ;D=0,T=1,W1 INDX<-W1 INDX-W2 INDX
        AND     #0FH                    ;W1 INDX MSD clear
        SEB     0,F16
        BCC     DIV1                    ;W1 INDX<W2 INDX?
        CLB     0,F16                   ;No
DIV1:   CLT                             ;T<-0
        SED                             ;Decimal mode
        JSR     SUB0                    ;W1<-W1-W2
        BCS     DIV2                    ;Borrow arise ?
        TST                             W1 ;Yes,then test W1
        BEQ     DIV3
        DEC     W1
DIV2:   CLD                             ;Binary mode
        RRF     W3+6
        CLC
        LDA     W3+6
        ADC     #10H
        STA     W3+6                    ;W3 LSD<-W3 LSD+1
        RRF     W3+6
        BCC     DIV1                    ;W3 LSD>=15 ?
ERROR:  SEB     0,F16                   ;0 Division error
        CLD                             ;Decimal mode clear
        RTS
;
DIV3:   JSR     ADD0                    ;W1<-W1+W2,add back
        LDA     W3+7
        CMP     #12
        BCS     DIV6
        INC     W3+7
```

```
;                                       ;W1 MSD<--W1 LSD<--W3 MSD<--W3 LSD
DIV4:    LDY    #4                      ;4 bits left rotate
DIV5:    LDX    #6
         CLC
DIV51:   ROL    W3,X
         DEX
         BNE    DIV51
         LDX    #6
         LDA    #7
DIV52:   ROL    W1,X
         DEX
         DEC    A
         BNE    DIV52
         DEY
         BNE    DIV5                    ;Rotate end ?
         INX
         SET                            ;T flag set
         AND    #0FH
         BRA    DIV1
;
DIV6:    LDA    W3+1
         AND    #0F0H
         BNE    DIV10
         BBS    0,F16,DIV7
         LDA    W1+7
         CMP    #12
         BCS    DIV10
DIV7:    INC    W1+7
         BBC    4,W1+7,DIV41
         CLB    4,W1+7
         CLB    0,F16
DIV41:
         BRA    DIV4
ERROR1:
         BRA    ERROR
;
DIV10:   BBS    0,F16,ERROR             ;Over flow error
         LDA    W1+7
         CMP    #13
         BCS    ERROR                   ;Under flow error
DIV12:   LDY    #7                      ;W1<-W3 copy
         LDX    #W1
         SET                            ;T<-1
DIV15:   LDA    16,X
         INX
         DEY
         BNE    DIV15
;
```

```
          CLT                                   ;T<-0
          JSR       INDX                        ;"0" condense
          CLD                                   ;Decimal mode clear
          CLC
          LDA       W1
          ADC       W2
          STA                                   W1 ;Get sign bit
          CLB       0,F16                       ;Normal return
          RTS
;*****************************************************
;
;         W1 <- W1*W2
;
;*****************************************************
;
MULT:
          JSR       BFITL                       ;W3 <- W1 copy,W1 clear
          CLC                                   ;C flag clear
          ADC       8,X                         ;W1_INDX <- W1_INDX+W2_INDX
          CLB       0,F16
          BBC       4,W1+7,MULT2                ;W1_INDX <- W_INDX+W2_INDX
          SEB       0,F16                       ;Yes
          CLB       4,W1+7
MULT2:
          CLT                                   ;T flag clear
MULT7:
          LDA       W3+6
          AND       #$0F
          BNE       MULT3
          INC       W3+7                    ;W1_MSD -> W1_LSD -> W3_MSD -> W3_LSD
MULT9:
          LDY       #4
MULT6:
          LDX       #0
          LDA       #7
          CLC
MULT4:
          ROR       W1,X
          INX
          DEC       A
          BNE       MULT4
          LDX       #1
          LDA       #6
MULT5:
          ROR       W3,X
          INX
          DEC       A
          BNE       MULT5
          DEY
          BNE       MULT6                       ;4 bits right rotate end ?
          LDA       W3+7
          CMP       #12
          BCC       MULT7
          BBS       0,F16,MULT8
          LDX       #W1+1                       ;W1 "0" test
```

```
MULT81:
          LDA       0,X
          BNE       MULT8
          INX
          CPX       #W1+7
          BNE       MULT81
          BRA       DIV12                       ;Get result (to division routine)
MULT8:
          LDA       W1+7
          BNE       MULT10
          BBC       0,F16,ERROR1                ;Overflow error
          CLB       0,F16
MULT10:
          DEC       W1+7
          LDA       W1+7
          AND       #$0F
          STA       W1+7
          BRA       MULT9
MULT3:
          DEC       W3+6
          SED
          JSR       ADD0
          BCC       MULT31
          INC       W1
MULT31:
          BRA       MULT2
;*******************************************************
;
;         W1 <- W1-W2
;
;*******************************************************
;
SUB0:
          SEC
          LDA       #6
          LDX       #W1+6
          SET                                   ;T flag set
SUB01:
          SBC       8,X
          DEX
          DEC       A
          BNE       SUB01
          CLT                                   ;T flag clear
          RTS
```

```
;*******************************************************
;
;       W1 <- W1+W2
;
;*******************************************************
;
ADD0:
        CLC
        LDA     #6
        LDX     #W1+6
        SET                                 ;T flag set
ADD01:
        ADC     8,X
        DEX
        DEC     A
        BNE     ADD01
        CLT                                 ;T flag clear
        RTS
*******************************************************
;
;       Following "0" condense,then modify INDX and data
;
;*******************************************************
;
INDX:
        TST     W1+7                        ;Test W1 INDX
        BEQ     INDX1
        LDA     W1+6
        AND     #$0F
        BNE     INDX1                       ;Valid data remain ?
        LDX     #W1                         ;No
        JSR     SFR4                        ;Condense to LSB direction
        DEC     W1+7                        ;Modify INDX
        BRA     INDX                        ;again
INDX1:
        RTS
;*******************************************************
;
;       W1 -> W3 ,and 0 -> W1
;
;*******************************************************
;
BFITL:
        CLT
        LDX     #W1
```

```
BFITL1:
          LDA       0,X
          STA       16,X
          LDA       #0
          STA       0,X
          INX
          CPX       #W1+7
          BNE       BFITL1
          LDM       #0,W3+7                    ;W3_INDX initial
          CLD                                  ;Decimal mode clear
          SET                                  ;T flag set
          RTS
;******************************************************
;
;         WN 4 bits left shift
;
;******************************************************
;
SFL4:
          LDA       #4
SFL41:
          ASL       6,X
          ROL       5,X
          ROL       4,X
          ROL       3,X
          ROL       2,X
          ROL       1,X
          DEC       A
          BNE       SFL41
          RTS
;******************************************************
;
;         WN 4 bits right shift
;
;******************************************************
;
SFR4:
          LDA       #4
SFR41:
          LSR       1,X
          ROR       2,X
          ROR       3,X
          ROR       4,X
          ROR       5,X
          ROR       6,X
          DEC A
          BNE SFR41
          RTS
```

```
;*****************************************************
;
;       Adjust INDX to W1 to W2
;
;*****************************************************
;
ADIX:
        LDA     W1+7
        CMP     W2+7
        BEQ     ADIX1
        BCC     ADIX23
ADIX22:
        LDX     #W2
ADIX2:
        LDA     1,X
        AND     #$F0
        BEQ     ADIX21
        CPX     #W1
        BEQ     ADIX51
        LDX     #W1
ADIX5:
        JSR     SFR4
        DEC     7,X
        BRA     ADIX
ADIX23:
        LDX     #W1
        BRA     ADIX2
ADIX51:
        LDX     #W2
        BRA     ADIX5
ADIX21:
        JSR     SFL4
        INC     7,X
        BRA     ADIX
ADIX1:
        RTS
;*****************************************************
;
;       Execute addition,subtraction
;
;*****************************************************
;
ADSB:
        SED                             ;Decimal mode set
        CLT                             ;T flag clear
        JSR     ADIX                    ;Adjust INDX
ADSB1:
        BBS     3,W2,ADSB6
        BBS     0,W1,ADSB7
        BBS     0,W2,ADSB10
```

```
ADSB9:
        JSR     ADD0                    ;W2 <- W1+W2
        BCC     ADD2                    ;C arise ?
        SEB     0,F16                   ;Yes
        JSR     SFR4                    ;1 digit right shift
        DEC     W1+7                    ;INDX -1
        BMI     ADD3                    ;FF ?
        CLC                             ;No
        LDA     W1+1
        ADC     #$10
        STA     W1+1
ADD2:
        CLB     0,F16
        JSR     INDX
ADD3:
        CLD                             ;Decimal mode clear
        RTS
ADSB7:
        BBS     0,W2,ADSB9
ADSB10:
        JSR     SUB0                    ;W1 <- W1-W2
        BCS     SUB2
        INX
SUB3:
        LDA     #$99
        SEC
        SBC     0,X
        STA     0,X                     ;Get 99-X
        INX
        CPX     #W1+7
        BNE     SUB3
        DEX
        SET
        LDY     #6                      ;C=1
SUB4:
        ADC     #0
        DEX
        DEY
        BNE     SUB4
        CLT                             ;Get 100's complement
        COM     W1                      ;Get sign bit in bit0
SUB2:
        CLB     0,F16
        JSR     INDX
        CLD                             ;Decimal mode clear
        RTS
ADSB6:
        BBS     0,W1,ADSB11
        BBS     0,W2,ADSB9
        BRA     ADSB10
ADSB11:
        BBS     0,W2,ADSB10
        BRA     ADSB9
```

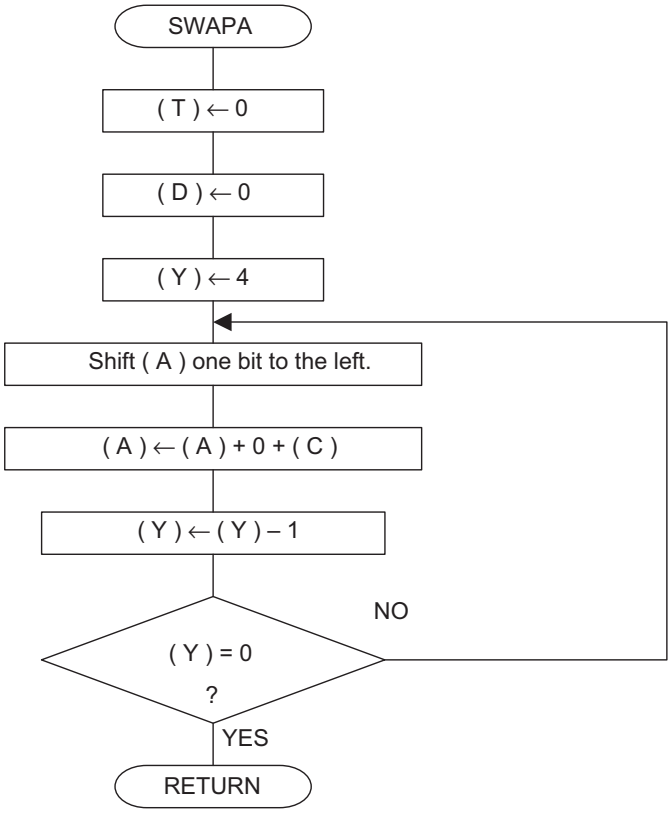
5. Substitute Instruction

5.1 Swap Accumulator

(1) Description

Accumulator A's upper four bits are swapped with the lower four bits.

(2) Flowchart



(3) Program List

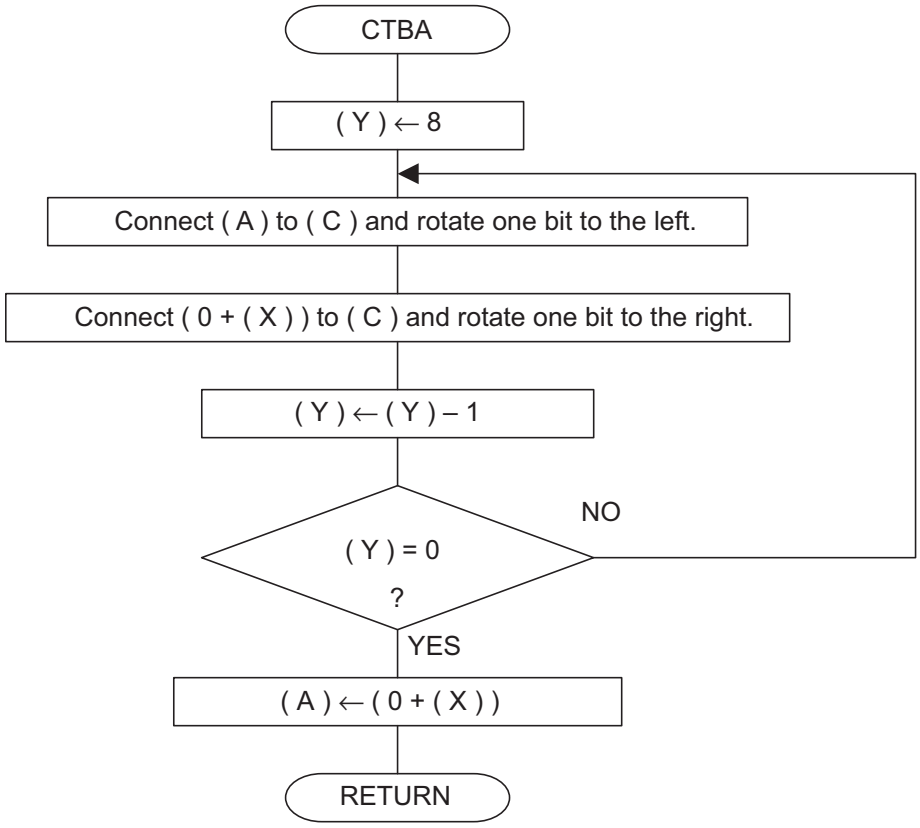
```
;*******************************************************
;
;         Swap A register
;
;*******************************************************
;
SWAPA:
          CLT                                   ;T flag clear
          CLD                                   ;Decimal mode clear
          LDY       #4
SWAPA1:
          ASL       A
          ADC       #0
          DEY
          BNE       SWAPA1
          RTS
```

5.2 Counter Bit Accumulator

(1) Description

The order of Accumulator A's bit data is reversed, using one byte of the work memory specified by Index Register X.

(2) Flowchart



(3) Program List

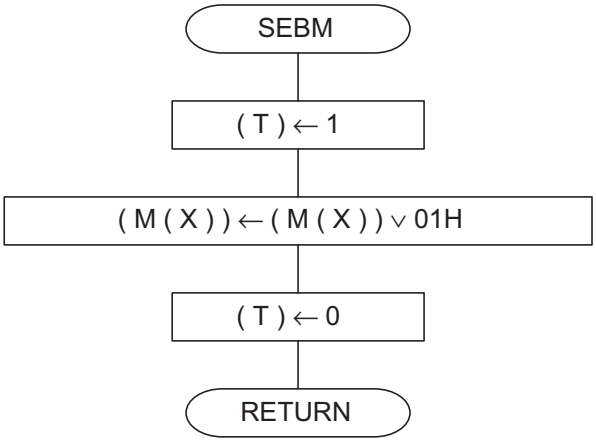
```
;*******************************************************
;
;         Counter bit A register
;
;*******************************************************
;
CTBA:
          LDY       #8
CTBA1:
          ROL       A
          ROR       0,X
          DEY
          BNE       CTBA1
          LDA       0,X
          RTS
```

5.3 Memory Set Bit

(1) Description

Bit 0 of the memory specified by Index Register X is set to "1".

(2) Flowchart



Note: M (X) represents the memory of the address indicated by Index Register X.

(3) Program List

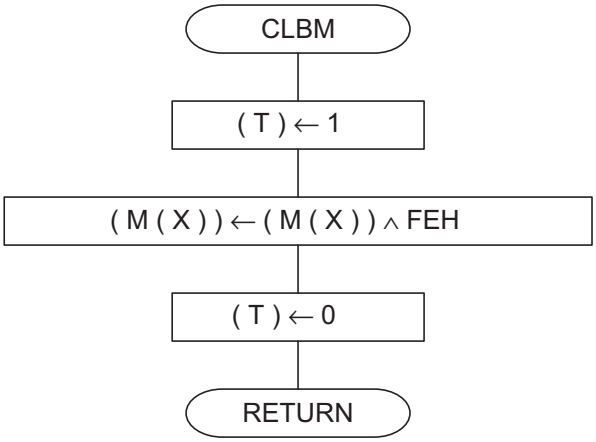
```
;******************************************************
;
;         Set bit 0 of M(X)
;
;******************************************************
;
SEBM:
          SET                                 ;T flag set
          ORA       #1                        ;Bit 0 set
          CLT                                 ;T flag clear
          RTS
```

5.4 Memory Clear Bit

(1) Description

Bit 0 of the memory specified by Index Register X is set to “0”.

(2) Flowchart



Note: M (X) represents the memory of the address indicated by Index Register X.

(3) Program List

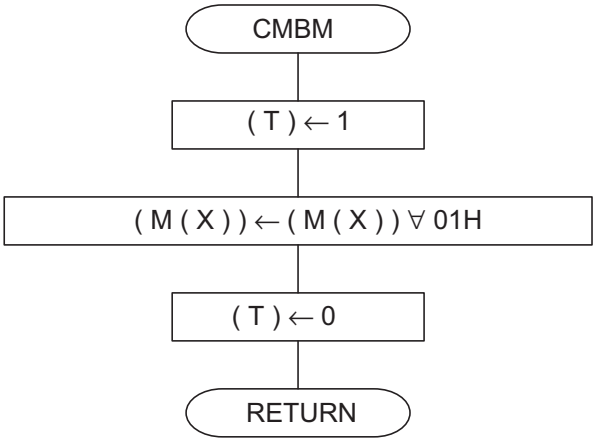
```
;******************************************************
;
;         Clear bit 0 of M(X)
;
;******************************************************
;
CLBM:
          SET                                 ;T flag set
          AND       #$FE                      ;Bit 0 clear
          CLT                                 ;T flag clear
          RTS
```

5.5 Memory Bit Reversal

(1) Description

Bit 0 of the memory specified by Index Register X is reversed.

(2) Flowchart



Note: M (X) represents the memory of the address indicated by Index Register X.

(3) Program List

```
;******************************************************
;
;         Complement bit 0 of M(X)
;
;******************************************************
;
CMBM:
          SET                                ;T flag set
          EOR       #1                       ;Bit 0 complement
          CLT                                ;T flag clear
          RTS
```

6. Program Usage Notes

The following usage notes apply to 740 Family products, in addition to all the instructions provided in the 740 Family Software Manual.

(1) The user must always perform thorough system evaluations.
(2) The program automatically handles X Modified Operation Mode Flag T and Decimal Mode Flag D as “0”. Do not call up this routine with either flag set to “1”.
(3) When executing the ADC or SBC instructions in the decimal mode (D = “1”), the processor status register’s Negative Flag N, Overflow Flag V and Zero Flag Z become invalid. In addition, Carry Flag C is set to “1” when the results of the operation generate a “carry”, and cleared to “0” when the results generate a “borrow”.
(4) In the decimal mode (D = “1”) and after execution of the ADC or SBC instruction, execute another instruction before execution of the SEC, CLC or CLD instruction.

7. Reference

Software Manual

740 Family Software Manual

Before using this material, please visit our website to verify that this is the most updated document available.

Renesas Technology Corporation Semiconductor Home Page

http://www.renesas.com

E-mail Support

E-mail: support_apl@renesas.com

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Feb.21.01	—	Issue as sample programs collection
1.01	Mar.18.05	—	Change to application note format and issue

Keep safety first in your circuit designs!

1. Renesas Technology Corp. puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.
   Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corp. product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corp. or a third party.
2. Renesas Technology Corp. assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corp. without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor for the latest product information before purchasing a product listed herein.
   The information described here may contain technical inaccuracies or typographical errors.
   Renesas Technology Corp. assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
   Please also pay attention to information published by Renesas Technology Corp. by various means, including the Renesas Technology Corp. Semiconductor home page (http://www.renesas.com).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corp. assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corp. semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corp. is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
   Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
8. Please contact Renesas Technology Corp. for further details on these materials or the products contained therein.